

Article

Collaborative Computation Offloading and Resource Management in Space–Air–Ground Integrated Networking: A Deep Reinforcement Learning Approach

Feixiang Li ^{1,*} , Kai Qu ¹, Mingzhe Liu ¹, Ning Li ¹ and Tian Sun ²

¹ The 15th Research Institute of China Electronics Technology Group Corporation, Beijing 100083, China; qukai@cetec.com.cn (K.Q.); liumingzhe@cetec.com.cn (M.L.); lining@cetec.com.cn (N.L.)

² Beijing Tsinghua Tongheng Urban Planning and Design Institute, Beijing 100085, China; suntian@thupdi.com

* Correspondence: lifeixiang126@126.com

Abstract: With the increasing dissemination of the Internet of Things and 5G, mobile edge computing has become a novel scheme to assist terminal devices in executing computation tasks. To elevate the coverage and computation capability of edge computing, a collaborative computation offloading and resource management architecture was proposed in space–air–ground integrated networking (SAGIN). In this manuscript, we established a novel model considering the computation offloading cost constraints of the communication, computing and cache model in the SAGIN. To be specific, the joint optimization problem of collaborative computation offloading and resource management was modeled as a mixed integer nonlinear programming problem. To address this issue, this paper proposed a computation offloading and resource allocation strategy based on deep reinforcement learning (DRL). Differing from traditional methods, DRL does not need a well-established formulation or previous information, and it is capable of revising the strategy adaptively according to the environment. The simulation results demonstrate the proposed approach can achieve the optimal reward values in the case of different terminal device numbers. Furthermore, this manuscript provided the analysis with variant parameters of the proposed approach.

Keywords: computation offloading; resource management; deep reinforcement learning; space–air–ground integrated networking



Citation: Li, F.; Qu, K.; Liu, M.; Li, N.; Sun, T. Collaborative Computation Offloading and Resource Management in Space–Air–Ground Integrated Networking: A Deep Reinforcement Learning Approach. *Electronics* **2024**, *13*, 1804. <https://doi.org/10.3390/electronics13101804>

Academic Editor: Carlo Mastroianni

Received: 14 April 2024

Revised: 30 April 2024

Accepted: 5 May 2024

Published: 7 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the advent of the Internet of Things era and the intelligent development of applications, the number of terminal devices (TDs) is growing rapidly. Due to the rapid growth of tasks, as well as the mobility and limited resources of terminal devices, complex computation tasks are difficult to execute on terminal devices promptly. Therefore, the terminal devices are capable of offloading the excessive computation task to the adjacent server that executes the task processing and returns the results, so as to solve the problem of insufficient computation capacity for terminal devices. Consequently, edge computing [1,2] has emerged.

In edge computing [3], terminal devices can obtain high-quality computation services through payment. However, the number of edge servers is limited. Once the number of offloading requests exceeds the threshold, the service quality of offloading will be affected. Scholars have conducted extensive research on the optimization of computation offloading performance [4]. Some are committed to optimizing the resource allocation of edge servers, and they usually establish the computation offloading problem as an optimization model [5]. Due to the limitations of wireless spectrum, there is a communication bottleneck between the terminal device and the road side unit, which affects the quality of computation offloading service. In addition, once there are too many requests for computation offloading services, edge servers may become overloaded, which can lead to computation offloading

failure or higher service latency. Therefore, some have researched how to expand the spectrum and computing resources of edge computing, e.g., unmanned aerial vehicles (UAV) [6,7], to build a multiple collaborative computation offloading mechanism. However, the network coverage and computation resources of edge computing are still greatly limited in the above research, and it is unable to provide ubiquitous computing services for terminals, especially in disaster areas, suburbs and other areas.

In order to enlarge the coverage and improve the ability of edge computing, the terminal device can offload computation tasks to the space–air–ground integrated networking (SAGIN) [8,9]. From Figure 1, as one of the most promising network architectures, SAGIN is a heterogeneous network based on ground networks and supplemented by space and air networks. It mainly includes computation nodes such as ground servers, UAVs, and low Earth orbit (LEO) satellites. SAGIN is expected to provide full coverage and high-quality computing services for terminal devices. However, the movement of terminal devices and computation nodes, as well as the uncertainty of channels, make SAGIN a time-varying network. How to effectively manage the resources of time-varying network is a challenging issue. In addition, SAGIN has large requirement for the time complexity of resource management algorithm. Traditional optimization method for solving the resource optimization problem in time-varying networks often require decoupling the original problem and repeating the solving process, leading to the interruption or failure of some delay-sensitive tasks.

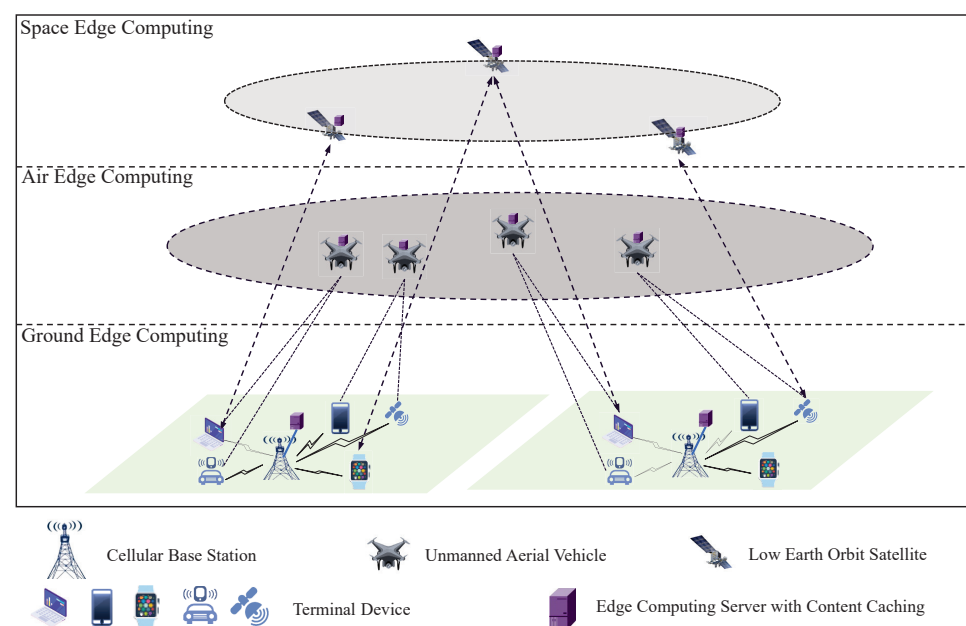


Figure 1. Architecture of edge computing in space–air–ground integrated networking.

Facing the shortcomings and challenges of existing research work, this manuscript considers computation offloading and resource management problem in the architecture of space–air–ground integrated network. In order to maximize the processing capability of SAGIN, we jointly established offloading decision, spectrum allocation, computation and storage resource scheduling as a mixed integer nonlinear programming (MINLP) problem. To address this issue, this manuscript proposes a computation offloading and resource management strategy relying on deep reinforcement learning (DRL) [10,11]. DRL has the advantage of adaptively rectification relying on the environment, while not requiring a well-established formulation or previous information. Therefore, DRL is capable of making decisions quickly when facing the time-varying SAGIN. The main contributions of this manuscript can be summarized as follows.

(1) A computation offloading architecture for SAGIN has been proposed, where the computation tasks of terminal devices can be processed locally or offloaded to ground

edge servers, air edge servers, or space edge servers. In addition, controllers empowered by deep reinforcement learning can make real-time computation offloading decisions and network resource allocation.

(2) In order to maximize the processing power of SAGIN, this paper jointly optimizes offloading decision, spectrum allocation, computation, and storage resource management in SAGIN as a MINLP problem. To address this issue, a computation offloading and resource allocation strategy based on deep reinforcement learning is proposed. Specifically, the offloading decision was continuously processed in this strategy, which enhancing the convergence of the networking.

The organization of this manuscript is summarized as following. Section 2 discusses the research work related to computation offloading and resource management. Section 3 provides the system model and formulates the optimization problem. Sections 4 and 5 details the algorithm for solving the problem. Section 6 evaluates the performance of the algorithm through simulation experiments. Finally, Section 7 concludes the manuscript.

2. Related Works

With the support of edge computing [12], terminal devices can provide an efficient computation and data processing service. However, once the number of offloading requests exceeds the threshold, the QoS of offloading will be affected. In response to this issue, scholars both domestically and internationally have conducted extensive research on optimizing computational offloading performance. For the sake of describing the difference between the current edge networks and our proposed approach, the main introduction will be provided in Table 1 and the following paragraphs.

From the perspective of this paper's authors, for the sake of improving the quality of computation offloading in edge computing [13], the following papers researched this problem from an economic perspective [14]. Zeng et al. [15] proposed an architecture cooperating with a volunteer terminal device and edge server, which was achieved by analyzing the optimal offloading data volume of terminal devices and the serve resource pricing of edge server through Stackelberg game theory. Differently, Zhang et al. [16] proposed a cloud–edge–end collaborative computation offloading mechanism and resource pricing strategy. Zhou et al. [17] proposed a novel optimization approach to solve multi-user computation offloading and resource management in edge computing. This solution aims to minimize energy consumption while considering latency limitations. Chen et al. [18] utilized the Deep Deterministic Policy Gradient algorithm to address the challenge of computation offloading and resource management. Different from optimizing a single performance metric, Gong et al. [19] proposed a joint optimization scheme for multiple IoT devices on account of deep reinforcement learning, aiming to minimize latency and energy consumption. Chen et al. [20] proposed a signal-based incentive mechanism that utilizes contract theory to address the information asymmetry issue in computation task offloading. They also tackled the D2D pairing problem in a many-to-many scenario. Peng et al. [21] jointly considered multi-user collaborative partial offloading, transmission scheduling, and computation allocation. An online resource coordination and allocation scheme was proposed to minimize latency and energy consumption. Fang et al. [22] considered a scenario where multiple users offload tasks to the same idle user. They investigated a multi-user computation task offloading problem, intending to maximize the overall efficiencies for all users in edge computing by jointly optimize channel allocation, device pairing, and offloading modes. The researches above deemed an isolated edge server as the provider of computing service. However, once there is a massive service request, the edge server may become overloaded, which leads to task interruption or failure.

Table 1. Comparison between current research and our proposed approach.

Reference	Approach	Scenario
[15]	Stackelberg game theory	Vehicular edge computing
[16]	Stackelberg game theory and genetic algorithm-based searching algorithm	Architecture of the vehicles , the edge servers, and the cloud
[17]	Value iteration-based reinforcement learning	Computation offloading and resource allocation in mobile edge computing
[18]	Deep deterministic policy gradient algorithm	Computation offloading and resource allocation in mobile edge computing
[19]	Deep reinforcement learning	Multi-access edge computing in Industrial Internet of Things
[20]	Signal-based incentive mechanism	Device-to-device computation offloading
[21]	Online resource coordination and allocation scheme	Device-to-device computation offloading
[22]	A potential game approach	Multiuser computing task offloading problem in device-enhanced MEC
[23]	Deep deterministic policy gradient algorithm	Multi-access edge computing and unmanned aerial vehicle
[24]	Soft actor critic algorithm	Drone-assisted multi-access edge computing
[25]	Deep reinforcement learning approach	Computation offloading and resource allocation in aerial to ground network
[26]	Primal decomposition approach	Air-to-ground communication and computation scenario
[27]	Edge-embedded lightweight algorithm	Distributed edge–cloud collaborative framework for UAV object detection
[28]	Q-learning based iterative algorithm	Task offloading to unmanned aerial vehicle swarm
[29]	Cooperative resource allocation approach	Computation-intensive Industrial Internet of Things applications
[30]	Difference-of-convex programming algorithm	Flexible deployment of UAVs
Our proposed approach	Deep reinforcement learning	Collaborative computation offloading and resource management in space–air–ground integrated networking

In an effort to expand the computing resources of edge networks, the authors of the following papers used drones to assist terminal devices in computing offloading. Peng et al. proposed a UAV-assisted edge computing networking architecture for computation offloading. This research intended to optimize computation offloading decisions and resource allocation jointly, to maximize the number of device computing tasks by DDPG algorithm [23]. Furthermore, a soft actor–critic algorithm was adopted to improve it [24]. Seid et al. [25] proposed a UAV-assisted emergency collaborative computation offloading and resource management method in air–ground networking, which considered limitations while

minimizing task latency and energy consumption sufficiently. Zhou et al. [26] investigated a UAV-oriented computation offloading problem for the sake of completing its computation task demands with the help of ground edge computing facilities. Yuan et al. [27] proposed a novel UAV edge and cloud collaborative framework for the sake of object detection, which attempts to achieve the moving targets in a timely and accurate manner. Ma et al. [28] designed a novel task-offloading framework that vehicles' computation jobs could execute natively; otherwise, they were offloaded to UAVs and ECDs. Liu et al. [29] established the processor resources and energy consumption model for the task-offloading approach. Dinh et al. [30] proposed a communication method that utilizes the flexible deployment of UAVs and their cooperative transmissions to improve accessing in the networking. Although the above manuscripts utilized adjacent vehicles, UAVs or cloud servers to reduce the load on the edge server, the coverage of in-edge networks is still limited and may not be able to provide a ubiquitous computation service for terminal device.

3. System Model

3.1. Architecture of Space–Air–Ground Integrated Networking

From Figure 2, the offloading scenarios in SAGIN are composed of a cellular base station in ground, an unmanned aerial vehicle in air, a low earth orbit satellite in space and terminal devices of users. In SAGIN, tasks of terminal devices can be processed locally or offloaded as a whole to the ground edge server, air edge server or space server for execution. For the sake of achieving the separation of control platform and data platform in this network architecture, SAGIN is divided into a ground layer, air layer and space layer based on Software Defined Networking architecture. with each layer managed by a controller. In addition, the delay in resource coordination between nodes in the same layer is not considered, and resources in different layers are considered independent of each other. TDs are able to choose different offloading computation mechanisms according to their respective demands. To be specific, in this network architecture there is three modes to offload scenarios by terminal devices, i.e., device-to-ground, device-to-air and device-to-space. In addition, the communication model of above will be provided as follows.

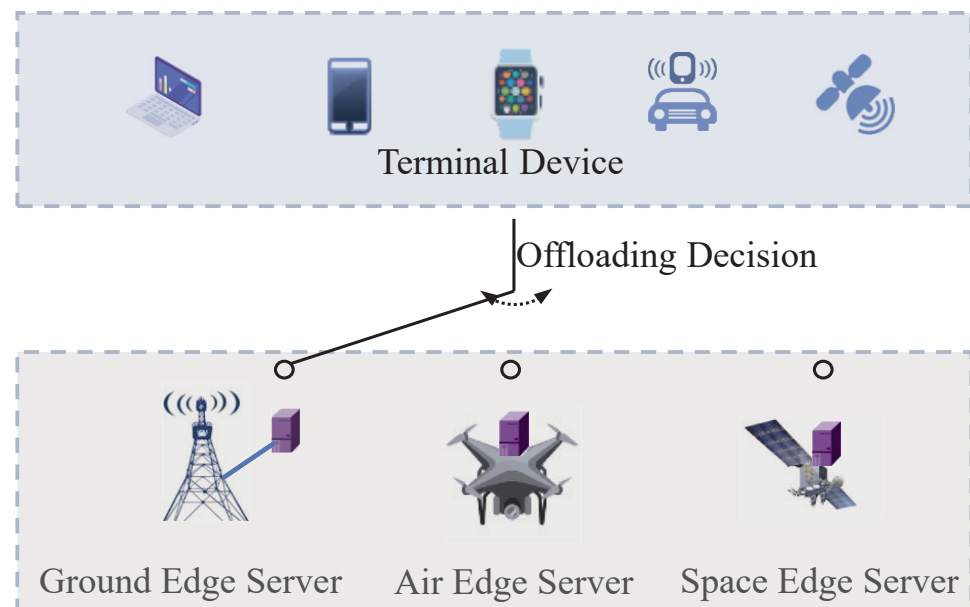


Figure 2. Offloading scenarios of edge computing in space–air–ground integrated networking.

3.2. Wireless Communication Model

(1) Task offloaded mechanism to ground. In this mechanism, w_{mg} is designated as the channel bandwidth between TD m ($m \in M$) and ground g and p_{mg} represents the transmission power. In addition, σ^2 is the constant additive noise power, $h_{m,g}$ represents the

channel gain and $k_{mg} \in \{0, 1\}$ is the interference factor. Accordingly, the data transmission rate r_{mg} in this mechanism can be established as

$$r_{mg} = w_{mg} \log_2 \left(1 + \frac{p_{mg} |h_{mg}|^2}{\sigma^2 + \sum_{m=1}^M k_{mg} p_{mg} |h_{mg}|^2} \right). \quad (1)$$

From the above model, assume that α_{mg} is designated as the access fee from TD m , charged by ground edge servers. β_{mg} represents the usage cost of spectrum, paid by ground edge servers. Thus, the communication income can be established as $R_{mg}^{comm} = \alpha_{mg} r_{mg} - \beta_{mg} w_{mg}$. Furthermore, $\alpha_{mg} r_{mg}$ means the income of ground edge cloud from user m , and $\beta_{mg} w_{mg}$ is defined as the cost of ground edge cloud to pay for the usage of bandwidth.

(2) Task offloaded mechanism to air. Similarly, the channel bandwidth is defined by w_{ma} and p_{ma} indicates the transmission power. To be specific, σ^2 is the constant additive noise power and $h_{m,a}$ expresses the channel gain. So, the data transmission rate r_{ma} is established as

$$r_{ma} = w_{ma} \log_2 \left(1 + \frac{p_{ma} |h_{ma}|^2}{\sigma^2 + \sum_{m=1}^M k_{ma} p_{ma} |h_{ma}|^2} \right), \quad (2)$$

where $k_{ma} \in [0, 1]$ means the interference between TD m and air edge server a .

Assume that α_{ma} indicates the access fee paid by TD m . The usage cost of spectrum is defined as β_{ma} , which is paid by air edge cloud a . The communication income can be established as $R_{ma}^{comm} = \alpha_{ma} r_{ma} - \beta_{ma} w_{ma}$. Moreover, $\alpha_{ma} r_{ma}$ denotes the revenue of air edge server a from user m , and $\beta_{ma} w_{ma}$ means the cost of air edge server a to defray for the usage of bandwidth.

(3) Task offloaded mechanism to space. Similar to the task offloading the mechanism to ground, the channel bandwidth is expressed by w_{ms} , and p_{ms} is the transmission power. Moreover, σ^2 stands for the constant additive noise power and $h_{m,s}$ denotes the channel gain. Accordingly, the data transmission rate r_{ms} is established as

$$r_{ms} = w_{ms} \log_2 \left(1 + \frac{p_{ms} |h_{ms}|^2}{\sigma^2 + \sum_{m=1}^M k_{ms} p_{ms} |h_{ms}|^2} \right), \quad (3)$$

where $k_{ms} \in [0, 1]$ means the interference between TD m and space edge server s .

Let α_{ms} denote the access fee paid by TD m . The usage cost of spectrum is expressed as β_{ms} , which is paid by space edge server s . The communication revenue can be established as $R_{ms}^{comm} = \alpha_{ms} r_{ms} - \beta_{ms} w_{ms}$. To be specific, $\alpha_{ms} r_{ms}$ denotes the revenue of space edge server s from user m , and $\beta_{ms} w_{ms}$ indicates the cost of space edge server s to defray for the usage of bandwidth.

3.3. Computation Model

In this scenario, TDs are expressed by $\mathbf{M} = \{1, 2, \dots, M\}$. $J_m = \{A_m, A'_m, B_m, T_m^{max}\}$, which represented the task from TD m . Furthermore, A_m and A'_m indicate the volume of the task J_m before and after computation. B_m denotes CPU number required by task J_m . For task J_m , T_m^{max} represents the maximum delay. Task J_m chooses the following different mechanisms base on its demand.

(1) Computation model in ground edge server. The computation delay by the ground edge server is formulated as $T_{mg}^c = \frac{B_m}{C_{mg}}$. To be specific, C_{mg} denotes the computation capacity of the ground edge server.

The computation rate of task m by the ground edge server is formulated as

$$q_{mg} = \frac{A_m}{B_m / C_{mg}} = \frac{A_m C_{mg}}{B_m}, \quad (4)$$

The computation energy consumption is expressed as $e_{mg} = \omega_{mg} T_{mg}^c$, where ω_{mg} stands for the energy consumption per second for ground edge server.

Suppose that ϕ_{m0} stands for the computation fee from user m , charged by ground edge server. Specifically, the ground edge server pays for the computation cost to compute the task at ground edge server indicated as ϕ_{mg} . The computation income model can be deemed as $R_{mg}^{comp} = \phi_{mg} q_{mg} - \phi_{mg} e_{mg}$. Furthermore, $\phi_{mg} q_{mg}$ represents the revenue of the ground edge server from user m , and $\phi_{mg} e_{mg}$ is the cost of ground edge server to defray for the usage of servers.

(2) Computation model in air edge server. In the same way, the computation delay formulated by air edge server n for task J_m is denoted as $T_{ma}^c = \frac{B_m}{C_{ma}}$, where C_{ma} is the computation capacity of air edge server a .

The computation rate of task m by air edge server a is established as

$$q_{ma} = \frac{A_m}{B_m/C_{ma}} = \frac{A_m C_{ma}}{B_m}, \quad (5)$$

Moreover, the computation energy consumption by air edge server a is indicated as: $e_{ma} = \omega_{ma} T_{ma}^c$, where ω_{ma} stands for the energy consumption for air edge server a .

Assume the computation cost charged by air edge server a is expressed as ϕ_{ma} . The computation cost ϕ_{ma} stands for computing the task at air edge server a . Eventually, the computation income model can be formulated as $R_{ma}^{comp} = \phi_{ma} q_{ma} - \phi_{ma} e_{ma}$. Moreover, $\phi_{ma} q_{ma}$ indicates the income of air edge server a from user m , while $\phi_{ma} e_{ma}$ is cost of air edge server a to disburse for the usage of servers.

(3) Computation model in space edge server. In the same way, the computation delay executed by space edge server s for task J_m denoted as $T_{ms}^c = \frac{B_m}{C_{ms}}$, where C_{ms} is the computation capacity by space edge server s .

The computation rate of task m by space edge server s is formulated as

$$q_{ms} = \frac{A_m}{B_m/C_{ms}} = \frac{A_m C_{ms}}{B_m}, \quad (6)$$

Specifically, the computation energy consumption by space edge server s is deemed as: $e_{ms} = \omega_{ms} T_{ms}^c$, where ω_{ms} is the energy consumption per second for space edge server s .

Suppose the computation fee charged by space edge server s is expressed as ϕ_{ms} . The computation cost ϕ_{ms} stands for computing the task at space edge server s . Finally, the computation income model can be formulated as $R_{ms}^{comp} = \phi_{ms} q_{ms} - \phi_{ms} e_{ms}$. Meanwhile, $\phi_{ms} q_{ms}$ indicates the income of space edge server s from user m , while $\phi_{ms} e_{ms}$ is cost of space edge server s to pay for the usage of servers.

3.4. Cache Model

In this model, we consider D contents are demanded in different edge servers. The caching strategy is decided by the parameter x' , $x' = 0$ denotes the content is not cached in the edge server, otherwise $x' = 1$ indicates the content is cached.

To be specific, the content widespread distribution is expressed as $\mathbf{G} = \{g_1, g_2, \dots, g_D\}$, where D is the maximal type number of content. Moreover, each TD asks for the content d with the probability g_d . Eventually, $\mathbf{G}$ submits to the Zipf distribution [31], and can be formulated as

$$g_d = \frac{1/d^\epsilon}{\sum_{d=1}^D 1/d^\epsilon}, \quad (7)$$

where the content widespread is characterized by ϵ , whose range is set as $[0.5, 1.5]$ [32]. Then, the gain is formulated as

$$l_{A'_m} = \frac{g_{A'_m} A'_m}{T_{A'_m}}, \quad (8)$$

where $T_{A'_m}$ is the download delay for cached content. In this manuscript, the price for caching the content has already been provided in the above. The backhaul cost is indicated as γ_{mg} , which is disbursed by ground edge server. Moreover, ψ_{ma} represents the storage fee to cache the content A'_m by ground edge server, which is charged by itself. In conclusion, the caching revenue of ground edge server can be formulated as $R_{ma}^{cache} = \psi_{ma}l_{A'_m} - \gamma_{ma}A'_m$. Specifically, $\psi_{ma}l_{A'_m}$ indicates the income of ground edge server from TD m , and $\gamma_{ma}A'_m$ is the cost of ground edge server to defray for the usage of backhaul bandwidth.

Let γ_{ma} stand for the backhaul cost of air edge server a . Moreover, the storage fee at air edge server a is denoted as ψ_{ma} . Therefore, the caching income of air edge server a can be computed as $R_{ma}^{cache} = \psi_{ma}l_{A'_m} - \gamma_{ma}A'_m$. Specifically, $\psi_{ma}l_{A'_m}$ is the revenue of air edge server n from TD m , and $\gamma_{ma}A'_m$ means the cost for air edge server a to defray to the usage of backhaul bandwidth.

In addition, the caching revenue of space edge server s can be computed as $R_{ms}^{cache} = \psi_{ms}l_{A'_m} - \gamma_{ms}A'_m$. Moreover, $\psi_{ms}l_{A'_m}$ is the income of space edge server s from TD m , and $\gamma_{ms}A'_m$ denotes the cost for space edge server s to disburse to the usage of backhaul bandwidth.

3.5. Problem Formulation

(1) Wireless access constraint. The allocated bandwidth frequency should not transcend the whole bandwidth of wireless link for three offloading mechanisms. To be specific, the maximal bandwidths of each scheme are assumed to be identical. Eventually, constraint 1 is expressed as $C1 : \sum_{m=1}^M w_{mg}F\{x_m = 1\} \leq w_{max}^{ground}$; constraint 2 is established as $C2 : \sum_{m=1}^M w_{ma}F\{x_m = 2\} \leq w_{max}^{air}$; and constraint 3 is deemed as $C3 : \sum_{m=1}^M w_{ms}F\{x_m = 3\} \leq w_{max}^{space}$.

(2) Computation resource constraint. The offloaded task should not exceed the maximal computational threshold of each edge server. Therefore, the constraints are established as follows. $C4 : \sum_{m=1}^M C_{mg}F\{x_m = 1\} \leq C_{max}^{ground}$, $C5 : \sum_{m=1}^M C_{ma}F\{x_m = 2\} \leq C_{max}^{air}$, $C6 : \sum_{m=1}^M C_{ms}F\{x_m = 3\} \leq C_{max}^{space}$.

(3) Energy consumption constraint. The energy consumption of the task should not surpass the maximal threshold. Therefore, the constraints are deemed as $C7 : \sum_{m=1}^M e_{mg}F\{x_m = 1\} \leq E_{max}^{ground}$, $C8 : \sum_{m=1}^M e_{ma}F\{x_m = 2\} \leq E_{max}^{air}$, $C9 : \sum_{m=1}^M e_{ms}F\{x_m = 3\} \leq E_{max}^{space}$.

(4) Computing offload scheme constraint. Considering that in this architecture the task of TD will be offload to ground node, air node and space node, then the offloading scheme is able to be established as $F\{x_m\} \in \{1, 2, 3\}$, where the value represents offloading to ground node, air node and space node, respectively.

(5) Problem formulation model. The target of this manuscript is to maximize the processing capability of SAGIN, which equals to maximize the computing offloading revenue. Considering offloading decision, communication, computation and spectrum resource, the TD computing offloading problem in SAGVN can be modeled as a multi-objective joint optimization (MINLP) problem. According to the above models and constraints, the problem formulation of computation offloading and resource management in SAGIN is modeled as

$$\begin{aligned}
& \max \sum_{g=1}^G \sum_{a=1}^A \sum_{s=1}^S (R_m^{comm} + R_m^{comp} + R_m^{cache}), \\
& \text{s.t. } C1 : \sum_{m=1}^M w_{mg} F\{x_m = 1\} \leq w_{max}^{ground}; \\
& \quad C2 : \sum_{m=1}^M w_{ma} F\{x_m = 2\} \leq w_{max}^{air}; \\
& \quad C3 : \sum_{m=1}^M w_{ms} F\{x_m = 3\} \leq w_{max}^{space}; \\
& \quad C4 : \sum_{m=1}^M C_{mg} F\{x_m = 1\} \leq C_{max}^{ground}; \\
& \quad C5 : \sum_{m=1}^M C_{ma} F\{x_m = 2\} \leq C_{max}^{air}; \\
& \quad C6 : \sum_{m=1}^M C_{ms} F\{x_m = 3\} \leq C_{max}^{space}; \\
& \quad C7 : \sum_{m=1}^M e_{mg} F\{x_m = 1\} \leq E_{max}^{ground}; \\
& \quad C8 : \sum_{m=1}^M e_{ma} F\{x_m = 2\} \leq E_{max}^{air}; \\
& \quad C9 : \sum_{m=1}^M e_{ms} F\{x_m = 3\} \leq E_{max}^{space}; \\
& \quad C10 : F\{x_m\} \in \{1, 2, 3\}.
\end{aligned} \tag{9}$$

4. Deep Reinforcement Learning Approach

4.1. Reinforcement Learning Algorithm

In the process of mutually engaging in the environment, reinforcement learning algorithm [33] attempts to seek the best behavior through trial and error. The optimal behavior not only pays attention to the immediate income, but also considers the income of the next n steps. The revenue of optimal behavior can be represented as

$$V^\pi(s_i) = r_i + \gamma r_{i+1} + \gamma r_{i+1}^2 + \dots, \tag{10}$$

Q learning algorithm is one of the representative reinforcement learning algorithm, which is value-based. The state behavior value, named the Q value, stands for the expectation of revenue when the action is adopted by a in the state s . To be specific, Q value is an evaluation value of the state and action, represented by immediate benefits and discounted benefits. It is formulated as

$$Q(s_i, a_i) \leftarrow r_i + \gamma V^\pi(s_{i+1}), \tag{11}$$

where $\gamma(0 < \gamma < 1)$ is the discount coefficient, which means the impact of future revenue on current behavior. The aim of Q learning algorithm is to maximize system utility. Let O_i substitute for r_i , and replace $V^\pi(s_{i+1})$ with $\max_{a_{i+1} \in A} Q(s_{i+1}, a_{i+1})$, then we can obtain

$$Q(s_i, a_i) \leftarrow O_i + \gamma \max_{a_{i+1} \in A} Q(s_{i+1}, a_{i+1}), \tag{12}$$

The exploration and utilization of strategy is a vital issue in reinforcement learning. To be specific, when the system state scale is tremendous, the method chosen for how to select the action effectively will affect the convergence speed and performance of the algorithm directly. Based on the behavior of Q value and $Index(s, a)$, this manuscript proposes a comprehensive action evaluation scheme to obtain a well-performed action.

When the system is at the state s_i , the algorithm will choose the action a_i with the following equation.

$$a_i \leftarrow \underset{a \in A}{argmax} (Q(s_i, a) + Index(s_i, a)), \tag{13}$$

where Q stands for evaluating the state and action. On the basis of Q value, the behavior index value is selected to maximize behavior benefit, which can be represented as

$$Index(s_i, a) = \zeta \sqrt{\frac{2 \ln n}{T_i(n)}} \min\{\frac{1}{4}, v_i(n)\}, \tag{14}$$

Meanwhile, ζ is a constant that is larger than zero. $T_i(n)$ represents the selected time of action a_i after n actions. $v_i(n)$ expresses the deviation factor that reflects the volatility, owing to introducing the variance of behavioral utility value $\sigma_i^2(n)$.

$$\sigma_i^2(n) = \frac{\sum_{t=1}^{T_i(n)} r_i^2(t)}{T_i(n) - O_i^2(T_i(n))}, \quad (15)$$

$$v_i(n) = \sigma_i^2(n) + \sqrt{\frac{2\ln n}{T_i(n)}}, \quad (16)$$

On the one hand, the behavior selection mechanism based on index $Index(s_i, a)$ gradually considers behavior with greater utility, reflecting the characteristics of utilization. On the other hand, if a behavior is not selected or is selected very few times with the iteration increases, then in the subsequent selection the behavior tends to be chosen, while reflecting the characteristics of exploration.

After the execution behavior is determined, the relay node executes behavior a_i . Moreover, the utility value O will be calculated and updated according to the following equation.

$$Q_{t+1}(s_i, a_i) = \begin{cases} (1 - \alpha)Q_i(s_i, a_i) + \alpha(O_i + \gamma \max Q_i(s_{i+1}, a_{i+1})), & s = s_i \text{ and } a = a_i \\ Q_i(s_i, a_i), & \text{otherwise} \end{cases}, \quad (17)$$

where $\alpha(0 < \alpha \leq 1)$ is a learning factor for state behavior, and it is formulated as $\alpha = \frac{1}{1+T_i(n)}$. The concrete executive process of Q-learning algorithm is shown as Algorithm 1.

Algorithm 1 Q-learning Algorithm

Input: state s , action a ;

Output: $Q(s, a)$;

Initialization:

Initialize behavior visit numbers $T_i(n) = 0$, state behavior value $Q(s_i, a_i) = 0$, state action query table ;

for $episode_1 = 1, I_1$ **do**

Initialize action vector $a = a_1, a_2, \dots$

As to the current state s_1

for $episode_2 = 1, I_{te}$ **do**

if $episode_2 = 1$ **then**

select a random action a_i ;

end if

if $episode_2 > 1$ **then**

$Index(s_i, a) \leftarrow \zeta \sqrt{\frac{2\ln n}{T_i(n)}} \min\{\frac{1}{4}, v_i(n)\}$;

According to $a_i \leftarrow \max_a [Q(s_i, a) + Index(s_i, a)]$ choose the action.

end if

Carry out action a_i in this mechanism, then obtain the reward value O_i and the next state value s_{i+1} ;

Compute $\alpha \leftarrow \frac{1}{1+T_i(n)}$

Update $Q(s_i, a_i)$

Update the state action query table;

end for

end for

In the following, we will provide the convergence analysis of this algorithm, the optimal Q value is represented as $Q^*(s_i, a_i)$.

Lemma 1. For the iteration of Q value with bounded revenue O , the learning factor is $\alpha(0 < \alpha \leq 1)$, and

$$\sum_{i=1}^{\infty} \alpha_{T_i(n)} = \infty, \sum_{i=1}^{\infty} \alpha_{T_i(n)}^2 < \infty, \forall s, a, \quad (18)$$

Then, when $T_i(n) \rightarrow \alpha, \forall s, a$, it can be obtained that

$$\lim_{i \rightarrow \infty} Q_i(s_i, a_i) = Q^*(s_i, a_i) \quad (19)$$

Proof. Firstly, the initial function is defined as $Q_0(s_i, a_i)$, then the optimal value $Q^*(s_i, a_i)$ can be updated with Equation (17) for $s_i \in S, a_i \in A$.

For functions $Q^*(s_i, a_i), O(s_i, a_i)$ and $Q_0(s_i, a_i)$, the constants $\varepsilon, \eta, \vartheta, \zeta$ and $\gamma (0 < \gamma < 1)$ should meet the following conditions.

$$\varepsilon O(s_i, a_i) \leq \gamma \max Q^*(s_{i+1}, a_{i+1}) \leq \eta O(s_i, a_i), \quad (20)$$

$$\vartheta Q^*(s_i, a_i) \leq Q_0(s_i, a_i) \leq Q^*(s_i, a_i), \quad (21)$$

where $0 < \varepsilon \leq \eta < \infty$ and $0 \leq \vartheta \leq \zeta < \infty$, owing to the optimal value is unknown, the values of $\varepsilon, \eta, \vartheta, \zeta$ can not be obtained directly. Therefore, we will verify that for the whole constants $\forall \varepsilon, \eta, \vartheta, \zeta, Q(s_i, a_i)$ is able to converge to the optimal solution after iterations.

If $0 \leq \vartheta \leq \zeta < 1$, then for $\forall i = 0, 1, \dots$, the reward function $Q_i(s_i, a_i)$ should satisfy the following conditions.

$$(1 + \frac{\vartheta - 1}{(1 + \eta^{-1})^i}) Q^*(s_i, a_i) \leq Q_i(s_i, a_i) \leq (1 + \frac{\zeta - 1}{(1 + \varepsilon^{-1})^i}) Q^*(s_i, a_i), \quad (22)$$

In addition, according to the mathematical induction method Equation (22) can be verified as follows.

When $i = 0$, it can be obtained that

$$\begin{aligned} Q_1(s_i, a_i) &= O(s_i, a_i) + \gamma \max Q_0(s_{i+1}, a_{i+1}) \\ &\geq O(s_i, a_i) + \vartheta \gamma \max Q^*(s_{i+1}, a_{i+1}) \\ &\geq (1 + \eta \frac{\vartheta - 1}{1 + \eta}) O(s_i, a_i) + \gamma (\vartheta - \frac{\vartheta - 1}{1 + \eta}) \max Q^*(s_{i+1}, a_{i+1}) \\ &= (1 + \eta \frac{\vartheta - 1}{1 + \eta}) [O(s_i, a_i) + \gamma \max Q^*(s_{i+1}, a_{i+1})] \\ &= (1 + \frac{\vartheta - 1}{1 + \eta^{-1}}) Q^*(s_i, a_i) \end{aligned} \quad (23)$$

Similarly, it can be inferred that

$$Q_1(s_i, a_i) \leq (1 + (\vartheta - \frac{1}{1 + \varepsilon^{-1}})) Q^*(s_i, a_i), \quad (24)$$

Then, when $i = 0$, Equation (22) can be satisfied.

Assuming that when $i = l - 1, l = 1, 2, \dots$, Equation (22) can also be satisfied. Then, when $i = l$, it can be acquired that

$$\begin{aligned} Q_1(s_i, a_i) &= O(s_i, a_i) + \gamma \max Q_0(s_{i+1}, a_{i+1}) \\ &\geq O(s_i, a_i) + \gamma (1 + \frac{\eta^{l-1}(\vartheta - 1)}{(1 + \eta)^{l-1}}) \max Q^*(s_{i+1}, a_{i+1}) \\ &\geq (1 + \eta^l \frac{\vartheta - 1}{(1 + \eta)^l}) [O(s_i, a_i) + \gamma \max Q^*(s_{i+1}, a_{i+1})] \\ &= (1 + \frac{\vartheta - 1}{(1 + \eta^{-1})^l}) Q^*(s_i, a_i), \end{aligned} \quad (25)$$

Similarly, it can be inferred that

$$Q_{1+1}(s_i, a_i) \leq (1 + \frac{\zeta - 1}{(1 + \varepsilon^{-1})^l}) Q^*(s_i, a_i), \quad (26)$$

Therefore, for $\forall i = 0, 1, 2, \dots$, Equation (22) can be satisfied.

In the following, we will prove that when $0 \leq \vartheta \leq 1 \leq \zeta < \infty$, the following equation is satisfied.

$$(1 + \frac{\vartheta - 1}{(1 + \eta^{-1})^l}) Q^*(s_i, a_i) \leq Q_i(s_i, a_i) \leq (1 + \frac{\vartheta - 1}{(1 + \eta^{-1})^l}) Q^*(s_i, a_i), \quad (27)$$

Let $i = 0$, and we can obtained that

$$\begin{aligned} Q_1(s_i, a_i) &= O(s_i, a_i) + \gamma \max Q_0(s_{i+1}, a_{i+1}) \\ &\leq O(s_i, a_i) + \zeta \gamma \max Q^*(s_{i+1}, a_{i+1}) + \\ &\quad \frac{\zeta - 1}{1 + \eta} [\eta O(s_i, a_i) - \gamma \max Q^*(s_{i+1}, a_{i+1})] \\ &= (1 + \frac{\zeta - 1}{1 + \eta^{-1}}) Q^*(s_i, a_i), \end{aligned} \quad (28)$$

Thus, when $0 \leq \vartheta \leq \zeta < \infty$ and for $\forall i = 0, 1, 2, \dots$, the reward function can satisfy Equation (22).

Eventually, due to the conclusion above, for random constant $\varepsilon, \eta, \vartheta, \zeta$, according to Equations (22)–(28), when $i \rightarrow \infty$, Equation (19) can be proved. $\square$

4.2. Deep Reinforcement Learning Algorithm

From Algorithm 2, the tack of deep Q-network is to exploit a feedforward neural network to approximate the Q-value function $Q(s, a; \theta)$. To be specific, the input layer of this Q-network is the state s , as well as the output layer is action a according to state s . The loss function measures the distance between the trained model and the actual model. The general goal is to minimize the loss function and continuously optimize the model. Furthermore, the parameter θ attempts to minimize the loss function with the following equation.

$$L(\theta) = E[(y(s, a, s'; \hat{\theta}) - Q(s, a; \theta))^2] \quad (29)$$

where the target function $y(s, a, s'; \hat{\theta}) = R + \lambda \max Q(s', a'; \hat{\theta})$ changes when the parameter $\hat{\theta}$ are updated.

Algorithm 2 Deep Q-learning Algorithm

Input: state s , action a ;

Output: $Q(s, a)$;

Initialization:

Initialize deep Q-network with weight θ ;

for $i < T$ **do**

if random probability $P < \delta$ **then**

 select an action a_i ;

otherwise

$a_i = \operatorname{argmax} Q(s, a; \theta)$;

end if

 Carry out action a_i in this mechanism, then obtain the reward R_i and the next state s_{i+1} ;

 Compute the target Q-value

$y(s, a, s'; \hat{\theta}) = R + \lambda \max Q(s', a'; \hat{\theta})$;

 Update deep Q-network by minimizing the loss value $L(\theta)$ with (29);

end for

5. Computation Offloading and Resource Management Approach Relying on Deep Reinforcement Learning

In the SAGIN scenario, the access channel conditions, computation capabilities and storage conditions are varying dynamically. It is a hassle to exploit a traditional mechanism to find an optimal solution properly. By comparison, DRL does not need a well-established formulation or previous information, while it can revise the strategy adaptively according to the environment. Thus, we exploit deep Q-learning to discover the optimal action efficiently, which is displayed in Figure 3. Furthermore, different strategies are determined according to offload task to space, air or ground. To be specific, if the offloaded task is cached, the caching model should be considered to economize the computation delay. Otherwise, the offloaded task will be calculated directly.

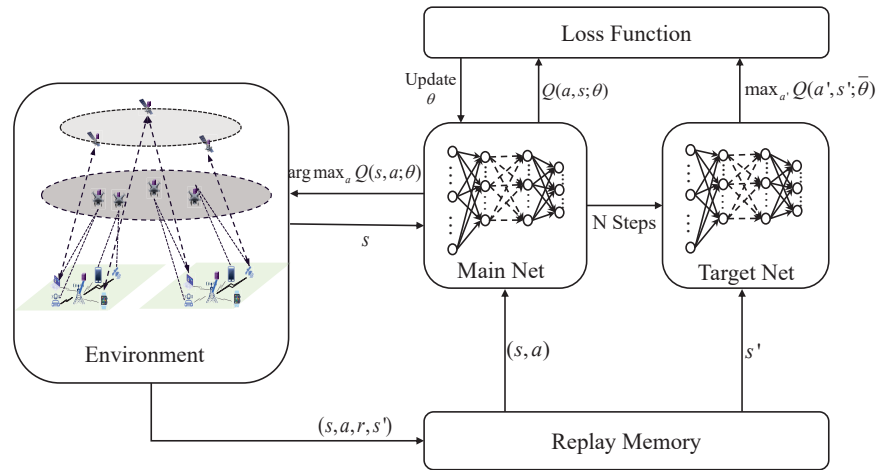


Figure 3. Deep reinforcement learning for computation offloading and resource management in SAGIN.

State Space. The state of the edge servers and the available cache $d \in \mathbf{D}$ for TD $m \in \mathbf{M}$ in timeslot t is determined by the random variable a_m^{comm} , a_m^{comp} and the random variable a_m^{cache} .

Action Space. In the SAGIN architecture, the agent should determine where offloaded task relying to the limited resource, whether or not the offloaded task has been cached in the server.

Correspondingly, the current action $a_m(t)$ at timeslot t is formulated as

$$a_m(t) = \{a_m^{comm}(t), a_m^{comp}(t), a_m^{cache}(t)\}, \quad (30)$$

where $a_m^{comm}(t)$, $a_m^{comp}(t)$, and $a_m^{cache}(t)$ are established as follows.

First, we define row vector $a_m^{comm}(t) = [a_{m,1}^{comm}(t), a_{m,2}^{comm}(t), \dots, a_{m,N}^{comm}(t)]$, where $a_{m,i}^{comm}(t)$, $i \in \{1, 2, \dots, N\}$ denotes whether TD m are connected to SAGIN edge server. Moreover, the value of $a_{m,i}^{comm}(t)$ is $\{1, 2, 3\}$, which indicates that at timeslot t the task in TD m choose to be offloaded to ground, air or space. Similarly, action $a_m^{comp}(t)$ is defined as $a_m^{comp}(t) = [a_{m,1}^{comp}(t), a_{m,2}^{comp}(t), \dots, a_{m,N}^{comp}(t)]$.

Then, row vector is defined as $a_m^{cache}(t) = [a_{m,1}^{cache}(t), a_{m,2}^{cache}(t), \dots, a_{m,N}^{cache}(t)]$, where $a_{m,j}^{cache}(t)$, $j \in \{1, 2, \dots, N\}$ indicates whether the content of TD m has been cached. Specifically, the value of $a_{m,j}^{cache}(t)$ is $\{0, 1\}$, where $a_{m,j}^{cache}(t) = 0$ represents that at timeslot t the content is not cached, otherwise $a_{m,j}^{cache}(t) = 1$ denotes cached.

Reward function. The reward of the edge server in SAGIN architecture is to jointly maximize the revenue of communication model, computation model and caching model. The reward function for TD m is formulated as follows.

$$R_m(t) = R_m^{comm}(t) + R_m^{comp}(t) + R_m^{cache}(t), \quad (31)$$

6. Simulation Result

In this section, the experiment environment of computation offloading and resource management approach relying on deep reinforcement learning in SAGIN will be introduced. We executed this approach Pycharm with python 3.6.13. The parameters of simulation are provided in the first subsection and the experimental results analysis will be provided in the second subsection. To be specific, the impact of the DNN's parameters is also discussed in the following subsection.

6.1. Parameter Setting

Based on Ref. [12] and the common setting in wireless networking, we set up the parameters' values in this experiment, as seen Table 2. In the following, the parameters of the deep neural network (DNN) will be described, which is a vital part of the Deep Q-learning mechanism. An input layer, two hidden layers, and an output layer constitute the network architecture of the DNN. In detail, the first hidden layer is set as 200 hidden neurons, and the second has 120. The output layer is equal to the number of TDs. To verify the effect of the TD number, the learning rate is 0.005 and memory size is 1024. Moreover, the training interval is set as 10, and training batch size is 128.

Table 2. Parameter settings.

System Parameters	Value Setting
The access fee charged by ground edge server	a random value in [1, 2] units/bps
The access fee charged by air edge server	a random value in [3, 4] units/bps
The access fee charged by space edge server	a random value in [5, 7] units/bps
The usage cost of spectrum paid by ground edge server	$[1 \times 10^{-4}, 2 \times 10^{-4}]$ units/Hz
The usage cost of spectrum paid by air edge server	$[3 \times 10^{-4}, 4 \times 10^{-4}]$ units/Hz
The usage cost of spectrum paid by space edge server	$[5 \times 10^{-4}, 7 \times 10^{-4}]$ units/Hz
The computation fee charged by ground edge server	0.2 units/J
The computation fee charged by air edge server	0.4 units/J
The computation fee charged by space edge server	0.6 units/J
The storage fee charged by ground edge server	10 units/byte
The storage fee charged by air edge server	15 units/byte
The storage fee charged by space edge server	20 units/byte

6.2. Simulation Result Discussion

The impact on simulation result by different parameters will be analyzed in the following experiment. Firstly, we will analyze the convergence of the proposed novel approach in SAGIN. Figures 4 and 5 depicts the changing normalized reward value and training loss rate by DQN with $TD_number = 20$. Figure 4 shows that the reward value increases rapidly with the increasing iteration, and then converge to the optimal value. From Figure 5, the training loss value decreases rapidly at the first 50 iteration and then remains stable. To be concluded, the proposed mechanism can seek the appropriated optimal solution owing on its strategy.

To verify the performance of DQN for computation offloading and resource management in SAGIN, different TD numbers are chosen. In this simulation, TD number is set as 5, 10, 20 in Figure 6, and it is obvious to obtain that the whole result curves can coverage to the optimal value finally. To be specific, with TD number increases, the reward value performs worse owing to the larger calculation quantity. At the case of $TD_number = 20$, the reward value could achieve a stable solution after 400 iterations, while at the case of $TD_number = 5$, the proposed approach is capable of achieving the optimal value at about iteration 100. To be concluded, with the increase in TD numbers, the signaling overhead raises apparently and the reward value of different TD numbers is affected subsequently.

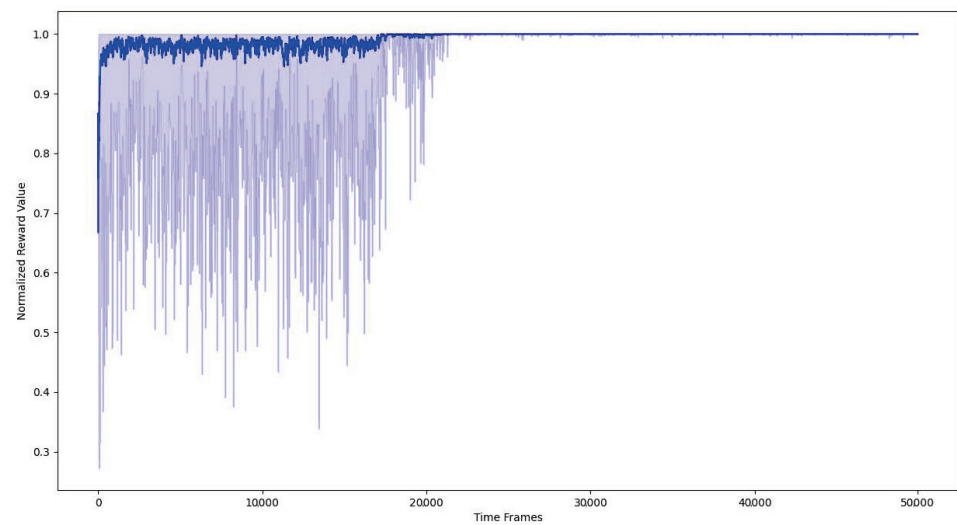


Figure 4. Convergence change of DQN for computation offloading and resource management in SAGIN.

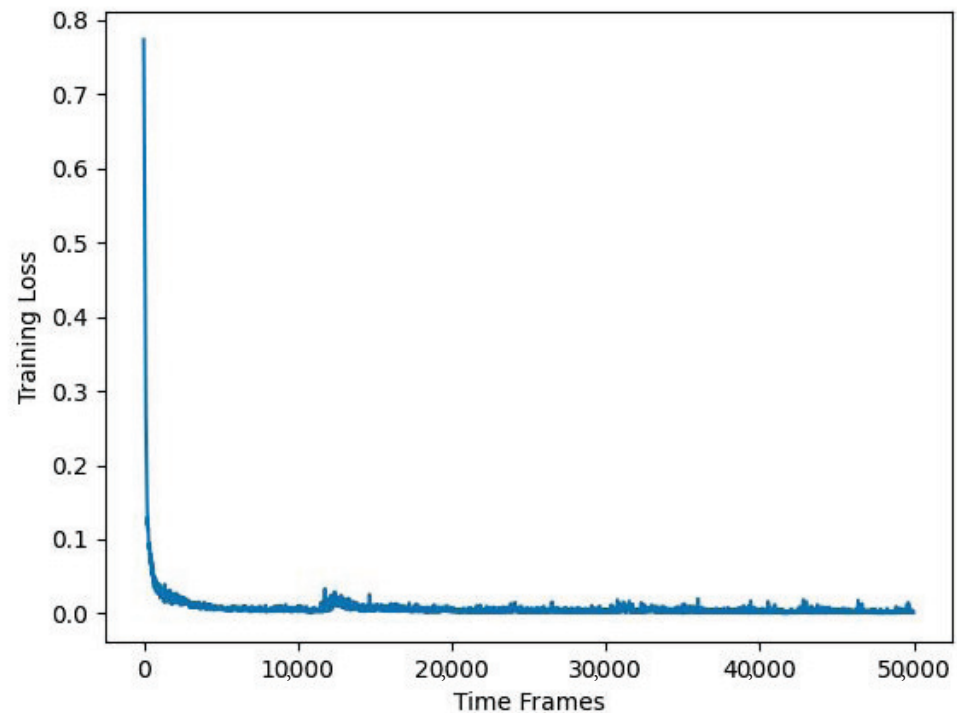


Figure 5. Training loss of DQN for computation offloading and resource management in SAGIN.

As can be seen in Figure 6, the proposed DQN approach is capable of converging to the optimum value rapidly in 100 iterations and finally obtaining the optimal reward value with different TD numbers. As shown in Figures 7–10, the proposed DQN approach with different parameters can achieve satisfying reward values ultimately. In the following, we will discuss the influence on experimental result by four critical parameters in DQN, i.e., learning rate, training interval, batch size and memory size. Learning rate dominates the learning process of the training model, which affects the convergence speed. Batch size has a great influence on the convergence degree and speed. A suitable batch size can prevent the algorithm from falling into local optimum. Similarly, training interval and memory size are also two vital parameters in DNN.

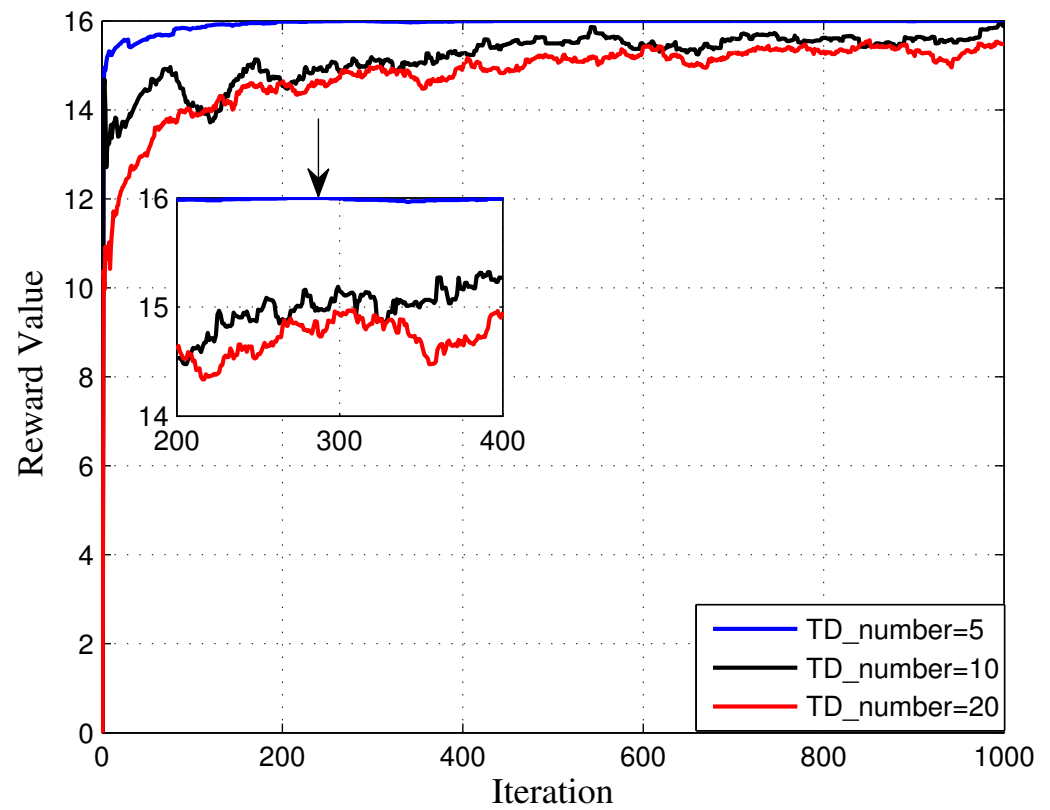


Figure 6. TD number impact on DQN-based mechanism in SAGIN.

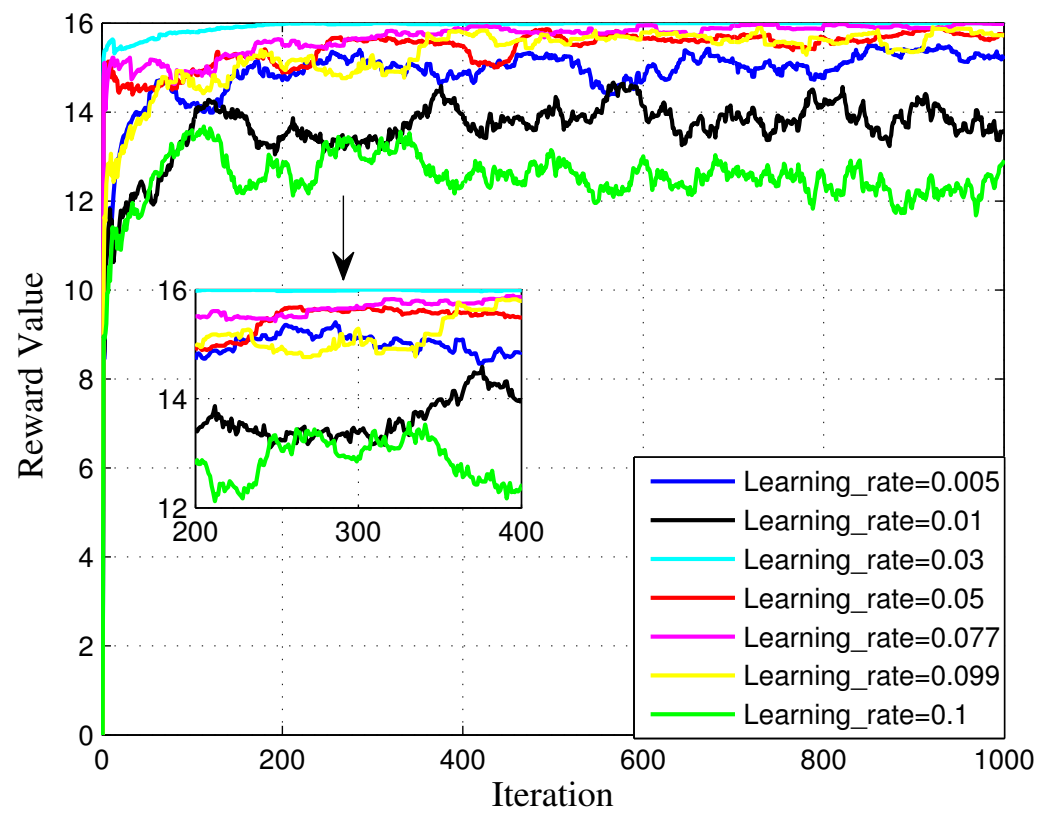


Figure 7. Learning rate impact on DQN-based mechanism in SAGIN.

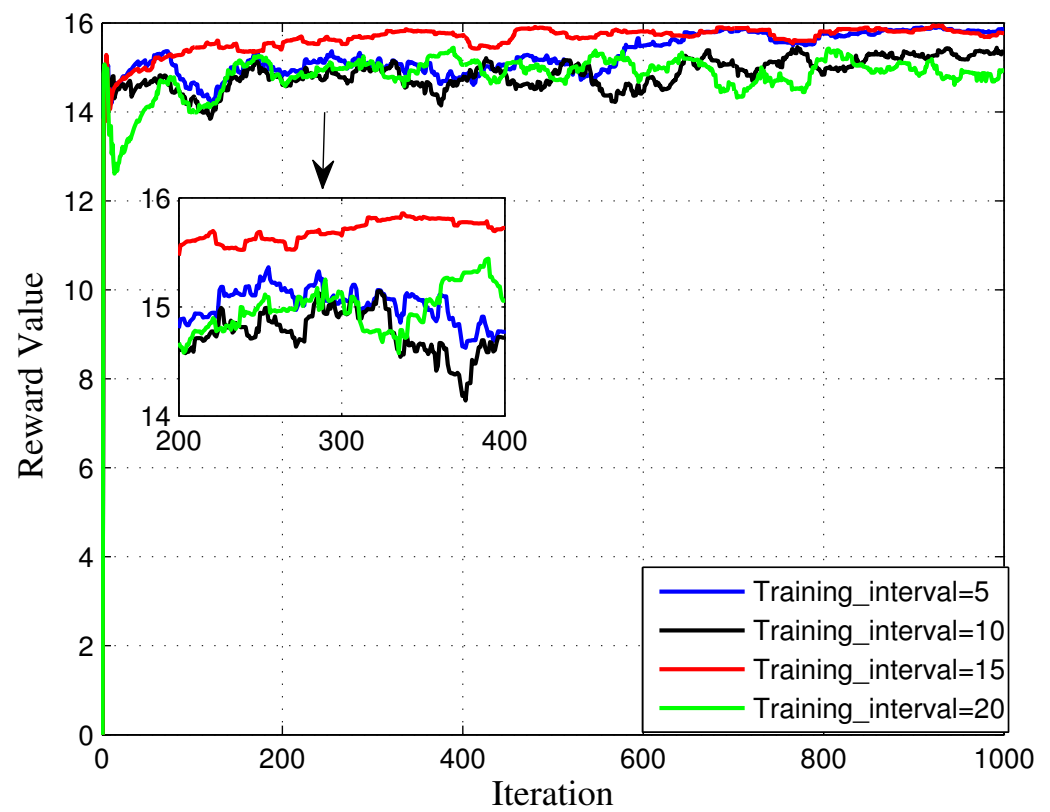


Figure 8. Training interval impact on DQN-based mechanism in SAGIN.

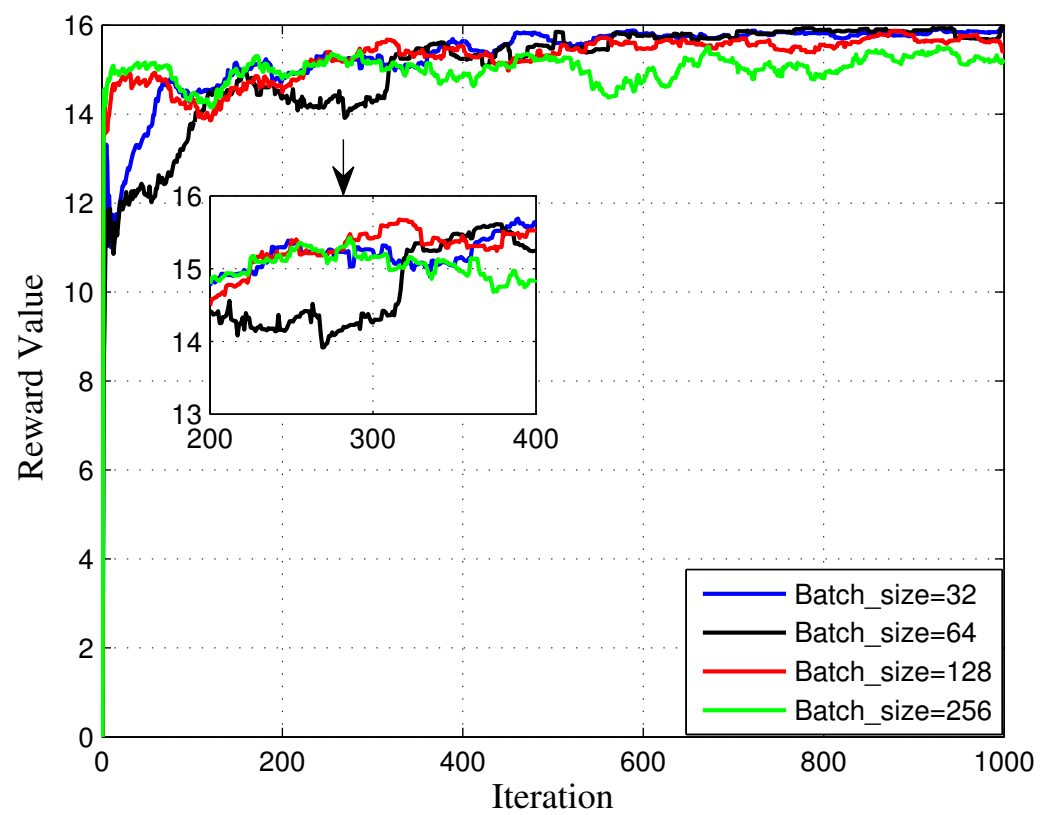


Figure 9. Batch size impact on DQN-based mechanism in SAGIN.

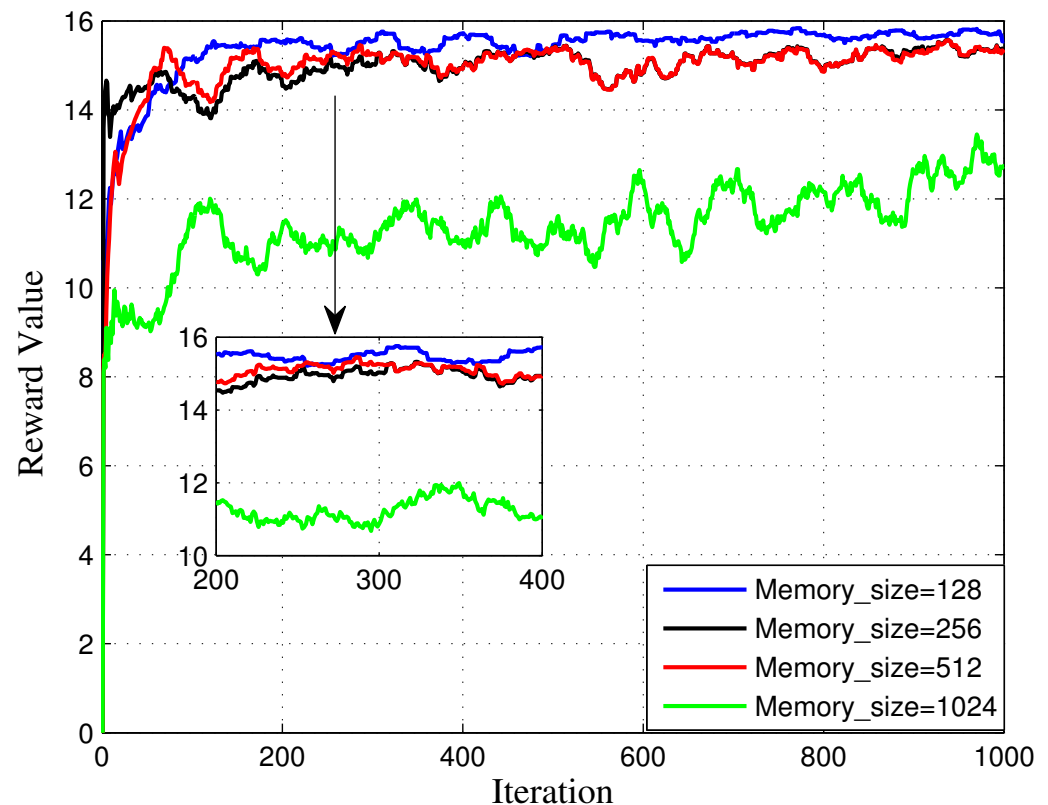


Figure 10. Memory size impact on DQN-based mechanism in SAGIN.

Figure 7 describes the impact on learning rate for DNN, which is set as 0.005, 0.01, 0.03, 0.05, 0.077, 0.099, 0.1. This figure demonstrates that learning rate has an important effect on the efficiency of the proposed approach. To be specific, in the case of *Learning rate* = 0.03, the reward value shown by DRL algorithm performs better than others during the whole iteration. *Learning rate* = 0.077 is not capable of completing the curve like *Learning rate* = 0.03, but surpasses other cases. To be specific, *Learning rate* = 0.1 has the worst performance among the different cases.

Figure 8 shows the reward value changes against iterations with different training intervals, while the value of training interval is determined as 5, 10, 15, 20. Obviously, we can obtain that while the training interval is 5 and 15, the reward value generated by DRL has a relatively better performance during the whole stage. In the case of *training interval* = 20, the reward value has a poor performance at the final stage.

As displayed in Figure 9, the batch rate has an influence in the simulation result, which is set as 32, 64, 128, 256. In detail, when batch size equals to 32, it achieves the best performance after 400 iterations, even compared to 64, 128, and 256. The reward value of *batch size* = 64 generated by DRL algorithm does not have a good performance at the first stage. Nevertheless, its reward value broadens swiftly and achieves a best performance when the iteration rate equals 1000. *Batch size* = 128 has a slightly worse performance than the case of *batch size* = 64. Additionally, *batch size* = 256 does not performing well.

Figure 10 provides the reward value changes against iterations with different memory size. Specifically, the memory size in the simulation experiment is set as 128, 256, 512, 1024. It is obvious to see that when memory size is set as 128, the reward value required by DQN performs best during the whole stage. Furthermore, when *memory size* = 1024, the reward value has a great distance with other three schemes.

To address the system performance issues, we provide the simulation results about the average delay and average normalized throughput among different TD numbers which is set as 5, 10, 20. Additionally, the numerical results of average delay time are 0.005550882577896118, 0.011090971231460571, 0.23159347009658812, as well as average normalized throughput are 0.9762736113164308, 0.9512185884374253, 0.8488317945596101,

respectively. From Figure 11, it can be achieved that when TD number equals to 20, the average delay time is largest and the average normalized throughput performs the worst. Through analysis, with the TD number increases, the whole transmission efficiency of this network system reduces correspondingly.

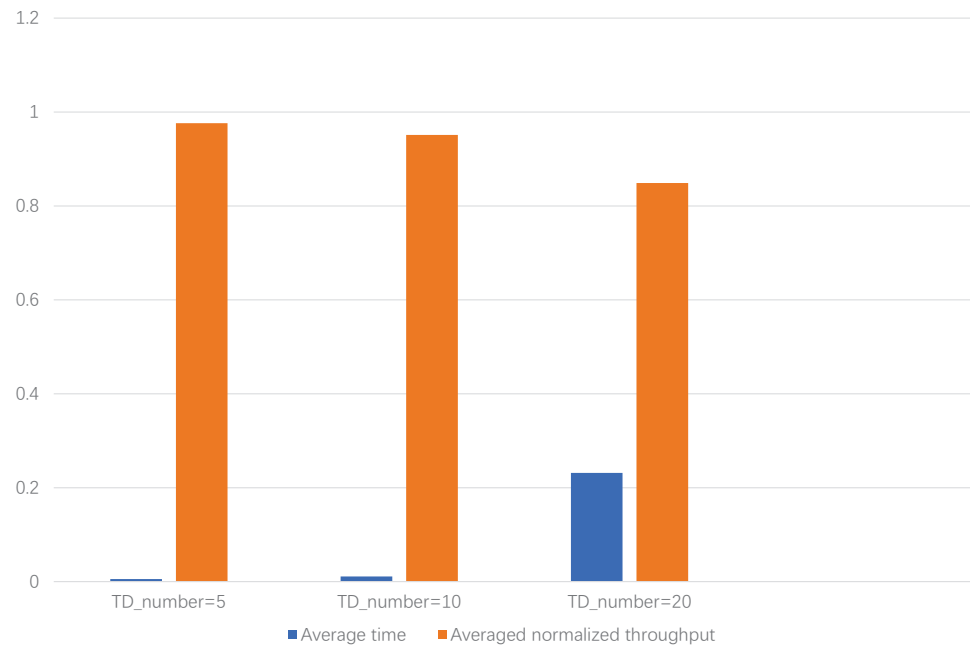


Figure 11. Analysis on average delay and average normalized throughput in SAGIN.

7. Conclusions

With the support of edge computing, terminal devices can offload the computation task to the edge server for obtaining high-quality computing services. However, the coverage range of existing edge networks was limited, which made it difficult to provide ubiquitous computing services for terminal devices. In order to further expand the coverage range of edge networks and improve their computation capability, this manuscript proposed a computation offloading and resource management architecture in SAGIN. In this architecture, in order to maximize the processing capability of SAGIN, the joint optimization problem of offloading decisions, spectrum, computation and storage resource management in SAGIN was modeled as an MINLP problem. To address this issue, this manuscript proposed a computational offloading and resource management mechanism relying on deep reinforcement learning. In this mechanism, the offloading decision was continuously processed, enhancing the convergence of the network. At the same time, a continuous reward function was designed to avoid high rewards caused by allocating too many resources to terminal devices. From the simulation results, the proposed DQN approach is capable of converging to the optimum value rapidly in 100 iterations and finally obtaining the optimal reward value with different TD numbers. Furthermore, the simulation results analyzed the impact of our proposed mechanism on different parameters and the proposed DQN approach with different parameters can achieve satisfying reward values ultimately.

Author Contributions: Conceptualization, F.L. and T.S.; methodology, K.Q.; software, M.L.; validation, N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is unavailable due to privacy.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Lin, H.; Zeadally, S.; Chen, Z.; Labiod, H.; Wang, L. A survey on computation offloading modeling for edge computing. *J. Netw. Comput. Appl.* **2020**, *169*, 102781. [\[CrossRef\]](#)
- Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [\[CrossRef\]](#)
- Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource scheduling in edge computing: A survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2131–2165. [\[CrossRef\]](#)
- Zhou, H.; Wang, Z.; Zheng, H.; He, S.; Dong, M. Cost Minimization-Oriented Computation Offloading and Service Caching in Mobile Cloud-Edge Computing: An A3C-Based Approach. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 1326–1338. [\[CrossRef\]](#)
- Li, K.; Wang, X.; He, Q.; Ni, Q.; Yang, M.; Dustdar, S. Computation Offloading for Tasks With Bound Constraints in Multiaccess Edge Computing. *IEEE Internet Things J.* **2023**, *10*, 15526–15536. [\[CrossRef\]](#)
- Dai, X.; Xiao, Z.; Jiang, H.; Lui, J.C.S. UAV-Assisted Task Offloading in Vehicular Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2023**, *23*, 2520–2534. [\[CrossRef\]](#)
- Subburaj, B.; Jayachandran, U.M.; Arumugham, V.; Suthanthira Amalraj, M.J.A. A Self-Adaptive Trajectory Optimization Algorithm Using Fuzzy Logic for Mobile Edge Computing System Assisted by Unmanned Aerial Vehicle. *Drones* **2023**, *7*, 266. [\[CrossRef\]](#)
- He, J.; Cheng, N.; Yin, Z.; Zhou, C.; Zhou, H.; Quan, W.; Lin, X.H. Service-Oriented Network Resource Orchestration in Space-Air-Ground Integrated Network. *IEEE Trans. Veh. Technol.* **2024**, *73*, 1162–1174. [\[CrossRef\]](#)
- Liu, L.; Li, C.; Zhao, Y. Machine Learning Based Interference Mitigation for Intelligent Air-to-Ground Internet of Things. *Electronics* **2023**, *12*, 248. [\[CrossRef\]](#)
- Han, D.; Mulyana, B.; Stankovic, V.; Cheng, S. A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation. *Sensors* **2023**, *23*, 3762. [\[CrossRef\]](#)
- Danino, T.; Ben-Shimol, Y.; Greenberg, S. Container Allocation in Cloud Environment Using Multi-Agent Deep Reinforcement Learning. *Electronics* **2023**, *12*, 2614. [\[CrossRef\]](#)
- Sadiki, A.; Bentahar, J.; Dssouli, R.; En-Nouaary, A.; Otrouk, H. Deep reinforcement learning for the computation offloading in MIMO-based Edge Computing. *Ad Hoc Netw.* **2023**, *141*, 103080. [\[CrossRef\]](#)
- Wu, G.; Wang, H.; Zhang, H.; Zhao, Y.; Yu, S.; Shen, S. Computation Offloading Method Using Stochastic Games for Software-Defined-Network-Based Multiagent Mobile Edge Computing. *IEEE Internet Things J.* **2023**, *10*, 17620–17634. [\[CrossRef\]](#)
- Li, F.; Fang, C.; Liu, M.; Li, N.; Sun, T. Intelligent Computation Offloading Mechanism with Content Cache in Mobile Edge Computing. *Electronics* **2023**, *12*, 1254. [\[CrossRef\]](#)
- Zeng, F.; Chen, Q.; Meng, L.; Wu, J. Volunteer Assisted Collaborative Offloading and Resource Allocation in Vehicular Edge Computing. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 3247–3257. [\[CrossRef\]](#)
- Zhang, Z.; Zeng, F. Efficient Task Allocation for Computation Offloading in Vehicular Edge Computing. *IEEE Internet Things J.* **2023**, *10*, 5595–5606. [\[CrossRef\]](#)
- Zhou, H.; Jiang, K.; Liu, X.; Li, X.; Leung, V.C. Deep Reinforcement Learning for Energy-efficient Computation Offloading in Mobile-edge-computing. *IEEE Internet Things J.* **2022**, *9*, 1517–1530. [\[CrossRef\]](#)
- Chen, J.; Huan, L.; Zhi, W.; Xu, L.; Tao, T. A DRL Agent for Jointly Optimizing Computation Offloading and Resource Allocation in MEC. *IEEE Internet Things J.* **2021**, *8*, 17508–17524. [\[CrossRef\]](#)
- Gong, Y.; Yao, H.; Wang, J.; Li, M.; Guo, S. Edge Intelligence Driven Joint Offloading and Resource Allocation for Future 6G Industrial Internet of Things. *IEEE Trans. Netw. Sci. Eng.* **2022**, early access. [\[CrossRef\]](#)
- Chen, M.; Wang, H.; Han, D.; Chu, X. Signaling-based incentive mechanism for d2d computation offloading. *IEEE Internet Things J.* **2022**, *9*, 4639–4649. [\[CrossRef\]](#)
- Peng, J.; Qiu, H.; Cai, J.; Xu, W.; Wang, J. D2d-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 4858–4873. [\[CrossRef\]](#)
- Fang, T.; Yuan, F.; Ao, L.; Chen, J. Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach. *IEEE Internet Things J.* **2022**, *9*, 3226–3237. [\[CrossRef\]](#)
- Peng, H.X.; Shen, X.S. DDPG-based Resource Management for MEC/UAV Assisted Vehicular Networks. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Victoria, BC, Canada, 16–18 December 2020; pp. 1–6.
- Peng, H.X.; Wu, H.; Shen, X.S. Edge Intelligence for Multi-dimensional Resource Management in Aerial Assisted Vehicular Networks. *IEEE Wirel. Commun.* **2021**, *28*, 59–65. [\[CrossRef\]](#)
- Seid, A.M.; Boateng, G.O.; Anokye, S.; Kwantwi, T.; Sun, G.; Liu, G. Collaborative Computation Offloading and Resource Management in Multi-UAV assisted IoT Networks: a deep reinforcement learning approach. *IEEE Wirel. Commun.* **2021**, *28*, 59–65.
- Zhou, J.; Tian, D.; Sheng, Z.; Duan, X.; Shen, X. Joint Mobility, Communication and Computation Optimization for UAVs in Air-Ground Cooperative Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 2493–2507. [\[CrossRef\]](#)
- Yuan, Y.; Gao, S.; Zhang, Z.; Wang, W.; Xu, Z.; Liu, Z. Edge-Cloud Collaborative UAV Object Detection: Edge-Embedded Lightweight Algorithm Design and Task Offloading Using Fuzzy Neural Network. *IEEE Trans. Cloud Comput.* **2024**, *12*, 306–318. [\[CrossRef\]](#)

28. Ma, X.; Su, Z.; Xu, Q.; Ying, B. Edge Computing and UAV Swarm Cooperative Task Offloading in Vehicular Networks. In Proceedings of the International Wireless Communications and Mobile Computing, Dubrovnik, Croatia, 30 May–3 June 2022; pp. 955–960.
29. Liu, J.; Li, G.; Huang, Q.; Bilal, M.; Xu, X.; Song, H. Cooperative Resource Allocation for Computation-Intensive IIoT Applications in Aerial Computing. *IEEE Internet Things J.* **2023**, *10*, 9295–9307. [[CrossRef](#)]
30. Dinh, P.; Nguyen, T.M.; Sharafeddine, S.; Assi, C. Joint Location and Beamforming Design for Cooperative UAVs With Limited Storage Capacity. *IEEE Trans. Commun.* **2019**, *67*, 8112–8123. [[CrossRef](#)]
31. Li, J.; Chen, H.; Chen, Y.; Lin, Z.; Vucetic, B.; Hanzo, L. Pricing and Resource Allocation via Game Theory for a Small-Cell Video Caching System. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 2115–2129. [[CrossRef](#)]
32. Jin, Y.; Wen, Y.; Westphal, C. Optimal Transcoding and Caching for Adaptive Streaming in Media Cloud: An Analytical Approach. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 1914–1925. [[CrossRef](#)]
33. Huang, L.; Bi, S.; Zhang, Y.J.A. Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Network. *IEEE Trans. Mob. Comput.* **2020**, *19*, 2581–2593. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.