# DRL-Based Long-Term Resource Planning for Task Offloading Policies in Multi-Server Edge Computing Networks

Haiyuan Li[1], Karcius D. R. Assis [1,2], Shuangyi Yan[1], and Dimitra Simeonidou[1]

***Abstract*—Multi-access edge computing (MEC) has been regarded as one of the essential technologies for mobile networks, by providing computing resources and services close to users, thereby, avoiding extra energy consumption and fitting the low-latency ultra-reliable requirements for emerging 5G applications. Task offloading policy plays a pivotal role in handling offloading requests and maximizing the network computing performance. Most recently developed offloading solutions are designed for instant rewards, therefore, neglecting the long-term computing resource optimization at the edge, which fail to deliver optimized network performance when a significant increase of computing requests appears. In this paper, with the objective of maximizing long-term offloading benefits on delay and energy consumption, task offloading policies are proposed to firstly avoid resource over-distribution through deep reinforcement learning (DRL) based resource reservation and server cooperation, and secondly maximize the average instant reward and the utilization of reserved resources by an optimization-based joint policy consisting of offloading decision, transmission power allocation and resource distribution. The DRL-based joint policy is evaluated in a simulated multi-server edge computing network. Compared to previous solutions, the DRL-based algorithms achieve higher and more reliable overall rewards. Of the implemented three DRL-based algorithms, fully cooperative multi-agent DRL accounts for cooperation between servers, achieving a 70.5% reduction in reward variance and a 13.4% increase in average rewards over 500 continuous operations. Resource balanced policies on long-term rewards help edge networks handle the explosive growth of 5G computing-intensive applications in the future.**

***Index Terms*—Multi-access network, task offloading, resource optimization, deep reinforcement learning, long-term scheduling policy**

## I. Introduction

RECENT advancements in internet of things (IoT) and mobile networks require a significant increase of computing resources in edge networks to support emerging applications that require ultra-low latency communication and intensive computing [1], [2]. In the fifth-generation (5G) networks, multi-access edge computing (MEC) has been introduced and received much attention from researchers and industries to alleviate computing pressure on edge devices. MEC allows edge devices to offload extensive workloads that are beyond their computing capabilities to nearby MEC servers [3], [4]. Effective offloading policies on decision and resource management for excessive offloading requests promisingly reduce the overall latency and energy consumption, and satisfy the availability requirement of 5G applications [5].

In previous studies, some of the task offloading algorithms assumed infinite computing resources on the MEC server for offloading operations [6], [7]. However, exploding computing-intensive tasks and the imbalanced request distribution are challenging the computing capability of MEC servers in bearer networks [8], [9]. The limited MEC resources require offload policies to consider long-term rewards and bring collaboration between multiple MECs. On the other side, to improve the utilization of limited MEC server resources, a large body of literature has focused on designing offloading resource management algorithms based on centralized [7], [10]–[12] and distributed approaches [8], [13], [14].

According to the number of servers, MEC systems in the current papers are divided into single-server [13], [15]–[17] or multiple-server [9]–[11], [18], [19] systems. Meanwhile, the techniques to tackle computation offloading and resource allocation in those models are classified into optimization-based [10], [11], [13], [19] or machined learning based [16], [17], [20], respectively.

Regarding the single-server system, Lyu *et al.* [13] introduced a semi-distributed approach that jointly optimizes offloading decisions and resource allocation problems. With a heuristic offloading algorithm (HODA), they achieved superior system utility with an acceptable complexity of $O(K^3)$. Nevertheless, the single-server system cannot deal with the coexistence between idle and over occupied servers because of the inevitable unbalanced request distribution [21], [22]. Under the same system model, Li *et al.* [16] designed a centralized-based DRL-based model to simultaneously resolve offloading decisions and computing resource allocations. However, excess users will cause the explosion of the action space of the DRL-based algorithm, which causes the model fail to converge. In comparison, Chen *et al.* [7] applied multi-agent policy-based DRL model that distributed the artificial intelligence (AI) agent to the edge devices to reduce the dimension of action space. However, this approach comes at the expense of complexity and might bring burdens on the devices because of their weak computing abilities.

In order to further improve the workload carrying capability of the MEC network, multiple access in RAN was introduced

Haiyuan Li, Shuangyi Yan and Dimitra Simeonidou are with the High Performance Networks Group, Smart Internet Lab, School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths (SCEEM), Faculty of Engineering, University of Bristol[1], BS8 1QU, U.K. (e-mail: ocean.h.li.2018@bristol.ac.uk).

Karcius Day R. Assis is with the Electrical and Computer Engineering Department, Federal University of Bahia (UFBA)[2], Salvador-BA, Brazil and also as a visitor at University of Bristol[1], U.K. (e-mail: karcius.assis@ufba.br)

Manuscript received XXXX YY, ZZZZ; revised YYYYY XX, ZZZZ.

as a promising technology to achieve the multi-server system. Li *et al.* [10], Nduwayezu *et al.* [12] and Xue *et al.* [11] explored the centralized joint optimization approach for the multi-access system. First fit (FF) algorithm [23] was used to select the offloading destination for the offloading requests in the network. In comparison, Apostolopoulos *et al.* proposed a distributed approach towards determining the optimal data offloading of each user within a multiple MEC servers system by non-cooperative game among the users. [8]. However, their assumption that all edge devices have access to the same set of servers neglected the geographical proximity factor in practical networks. In contrast, Kan *et al.* [24] designed a model where edge devices can only connect to their proximate servers. The multi-server system was achieved by relaying the workload among servers via the wired interface (Mp3) which is designed and standardized for the workload transition between MEC servers by the European Telecommunications Standards Institute (ETSI) [25]. Furthermore, Qian *et al.* designed a distributed algorithm and a centralized online DRL-based solution for statistic and dynamic channel scenarios, respectively, which can realized the selection of access point with minimized transmission cost [14].

Although extensive research has been carried out on the offloading management technology in single and multiple server scenarios, few studies have been reported with consideration of long-term resource balance for offloading operations. Offloading without long-term resource balance will prefer instant high reward at the expense of the resource shorting for following offloading requests.

In summary, three critical issues need to be resolved in designing efficient offloading management policies. First, the prior joint offloading algorithms in multi-server systems ignored the geographic proximity between edge devices and servers. FF-based solutions did not fundamentally deal with the unbalanced traffic distribution. Second, the possible action space explosion in DRL-based solutions remained unclear. Last but not least, the lack of consideration of long-term reward in the current studies hindered further improvement in network resource utilization. To overcome these obstacles, the main contributions of this paper are as follows:

- *Combination of DRL and Optimization-based Method:* With the objective of maximizing long-term offloading benefits on delay and energy consumption, task offloading policies are proposed to firstly avoid resource over-distribution through deep reinforcement learning (DRL) based resource reservation and server cooperation, and secondly maximize the average instant reward and the utilization of reserved resources by an optimization-based joint policy.
- *Problem Formulation:* The optimization problem is formed as an MINLP and decomposed into two interrelated subproblems. Solution of the first subproblem provides a long-term policy for resource scheduling and server cooperation, to reserve the accessible resource and avoid the impairment of resource over-occupation on future incoming requests. Multiple connected MEC servers form the offloading environment, in which the Mp3 wired interface technology is adopted to realize server cooperation and achieve resource balance.
- *Long-term Scheduling Principle:* The offloading process is modeled as a Markov decision process (MDP) to account for the impact of previous offloading actions on later operations. Three DRL algorithms, including a single-agent value-based deep Q network (DQN) model and two multi-agent policy-based deterministic deep policy gradient (MADDPG) models, are developed as the solutions to this subproblem. To facilitate the implementation of these DRL models in future 5G applications, the placement of the DRL agent is also discussed in the paper.
- *Short-term Utilization Maximization:* Taking the resolved accessible resources from the first subproblem as the upper limit, a joint policy, including offloading decision, computing and transmission resource allocation (DCTRA), is designed based on the optimization-based approach. This joint policy maximizes the utilization of reserved resources and the average instant reward of the requests within each server. As interrelating information, the results of the DCTRA are fed back to the DRL algorithms in the first subproblem.

To the best of our knowledge, this is the first time that the optimization-based method and DRL have been geared together to solve two interrelated subproblems and find the policies that maximize long-term offloading benefits. Regarding the joint solution between edge devices in one time slot, our results show that there is a great benefit in the execution time by sorting the execution order of requests and reusing the released resources for the on processing workloads. In addition, a period of continuous time slots is taken to illustrate the long-term reward of random allocation, over allocation, and three DRL-based allocation policies. We prove that over-allocation used by the prior studies can not get reliable results as it pursues higher rewards at the expense of future loss. In comparison, DRL-based algorithms can adapt to the diversity of future network states and obtain resource allocation strategies with better performance. Furthermore, of these three DRL algorithms, compared to single-agent DQN, the cooperative multi-agent model is able to more accurately account for interactions between servers on limited resources and achieve higher average rewards. In the end, the value of server cooperation is also justified.

The remainder of this paper is organized as follows. Section II presents the multi-server system scenario and formulates the optimization problem. Section III describes the proposed optimization-based and DRL integration solution. Then, section IV provides the setup and the numerical results. Finally, sections V concludes the paper.

## II. Task Offloading Scenario in Edge Computing Networks

The multi-server edge computing network supporting radio access networks (RANs) for 5G is presented in Figure 1. The radio units (RUs) are deployed close to the edge devices. Each RU can access multiple MEC servers through the distributed unit (DU) and core unit (CU) in RAN [26]. It is worth noting that the selection of proximity MEC servers for offloading

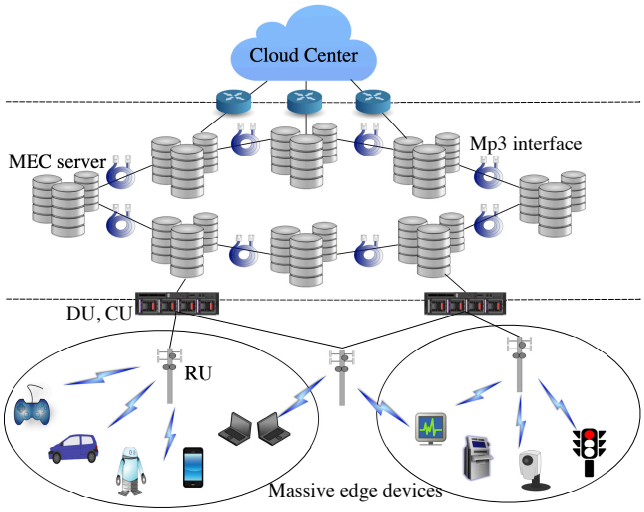



**Fig. 1:** MEC Network Architecture.

requests is done locally, and each server only handles requests that are sent to itself [24] [27]. With offloading policy, MEC servers can decide to redirect workload to other connected servers through the wired interface Mp3 [25].

In this multi-server system, each server serves a certain number of mobile devices that randomly send offload requests at any time slot. These requests are looking for more computing resources from servers to reduce their local processing energy consumption and execution delay.

In order to formulate the long-term optimization problem and quantify the benefits of task offloading policies, we model the execution delay and energy consumption of both local and edge computing in this section.

### A. System model

Within the MEC network, the offloading request from the mobile devices includes two parameters, namely $S_i$ and $C_i$. $S_i$ specifies the necessary data size for the task $i$ to be processed on the server, and $C_i$ represents the required amount of CPU (Cycles), i.e. computing resources to accomplish the task $i$. For ease of reference, important notations used throughout this paper are summarized in Table 1.

*1) Local Computing:* The local computing capability of mobile device with task $i$ is represented by $f_i^l$. With required CPU $C_i$, the local execution time can be written as

$$T_i^l = \frac{C_i}{f_i^l} \tag{1}$$

According to [28], the CPU energy consumption per cycle is determined by the execution frequency $f_i^l$ and can be written as $\kappa(f_i^l)^2$. $\kappa$ denotes the energy coefficient depending on the hardware property, typically $\kappa = 10^{-11}$. Therefore, the local energy consumption can be obtained as

$$E_i^l = \kappa \left(f_i^l\right)^2 C_i \tag{2}$$

*2) Offload Computing:* Compared to processing workloads locally, offloading them to the MEC servers and data center can greatly reduce the execution delay and convert local computing to less energy-intensive signal transmissions [29]. Within the model, the overhead of the back-load stage is ignored as the size of the output is generally much smaller than the input [15], [28].

**TABLE I:** Notations Used Throughout the Paper

| | Notation | Description |
|---|---|---|
| Index | $i$ | Task |
| | $j$ | Server |
| | $z$ | Time slot |
| Parameter | $\beta_i^e$ | Energy preference of $i$ |
| | $\beta_i^t$ | Latency preference of $i$ |
| | $C_i$ | Required computing resources of $i$ |
| | $E_i^l$ | Local energy consumption of $i$ |
| | $f_i^l$ | Local computing resources of $i$ |
| | $h_{ij}$ | Channel gain between $i$ and $j$ |
| | $\kappa$ | hardware energy coefficient |
| | $L$ | Time scale |
| | $N_0$ | Noise power |
| | $p_0$ | Maximum transmission power |
| | $Q_z$ | Total amount of requests at $z$ |
| | $Q_{jz}$ | Number of requests on $j$ at $z$ |
| | $Q_{jz}^d$ | Offloaded tasks on $j$ at $z$ |
| | $\gamma$ | Discount factor of DRL |
| | $S_i$ | Data size of $i$ |
| | $\tau$ | Target network update coefficient |
| | $\mu$ | Learning rate of DRL |
| | $U$ | Number of servers |
| | $W$ | User bandwidth |
| | $X_z$ | Remaining computing resources of $j$ at $z$ |
| | $\zeta$ | Power amplifier efficiency |
| Variable | $d_{ij}$ | Offloading decision of $i$ to $j$ |
| | $G_{jz}$ | Accessible computing resources for $Q_{jz}$ |
| | $K_{ij}$ | Data rate of $i$ to $j$ |
| | $M_{ij}$ | Distributed resources for $i$ from $j$ |
| | $O_{jz}$ | Occupation time of $G_{jz}$ |
| | $p_{ij}$ | Transmission power of $i$ to $j$ |
| | $T_{ij}^e$ | Remote execution time of $i$ on $j$ |
| | $T_i^l$ | Local computing time of $i$ |
| | $T_{ij}^r$ | Overall offloading delay of $i$ on $j$ |
| | $T_{ij}^t$ | Transmission time between $i$ and $j$ |
| | $R_{jz}$ | Average offloading reward of $Q_{jz}$ |
| Policy | $\mathcal{D}$ | Offloading decision |
| | $\mathcal{M}$ | Resource allocation between devices |
| | $\mathcal{P}$ | Transmission power allocation |
| | $\mathcal{G}$ | Long-term accessible resource allocation |

For offloading to the MEC server, the overall latency for task $i$ includes the transmission delay $T_{ij}^t$ to the server $j$ and the remote execution time $T_{ij}^e$. It can be written as

$$T_{ij}^r\left(M_{ij}, p_{ij}\right) = T_{ij}^t\left(p_{ij}\right) + T_{ij}^e\left(M_{ij}\right) \tag{3}$$

where $p_{ij}$ denotes the transmission power and $M_{ij}$ denotes the distributed computing resources for task $i$ on server $j$. According to [30], data rate on the fronthaul can be represented by

$$K_{ij}\left(p_{ij}\right) = W \log_2\left(1 + a_{ij} N_0\right) \tag{4}$$

in which $a_{ij} = h_{ij}/N_0$. $K_{ij}$ is effected by channel gain $h_{ij}$, transmission power $p_{ij}$ and ambient noise $N_0$. Based on data rate $K_{ij}$ and workload size $S_i$, $T_{ij}^t$ can be given as

$$T_{ij}^t\left(p_i\right) = \frac{S_i}{K_{ij}} \tag{5}$$

The other component, remote execution time $T_{ij}^{e}$ can be written as

$$T_{ij}^{e}\left(M_{ij}\right)=\frac{C_i}{M_{ij}} \tag{6}$$

As we have transmission power $p_{ij}$ and transmission time $T_{ij}^{t}$, the energy consumption by transmitting $i$ to $j$ can be calculated by

$$E_{ij}^{r}\left(p_{ij}\right)=\frac{p_{ij}}{\zeta}\cdot T_{ij}^{t}\left(p_{ij}\right)=\frac{p_{ij}}{\zeta}\frac{S_i}{W\log_2\left(1+a_{ij}N_0\right)} \tag{7}$$

where $\zeta$ is the power amplifier efficiency.

For transmission to the data center, a task $i$ needs to get access to a MEC server $j$ first, therefore, consumes the same device transmission energy as calculated in Function 7. Compared to the MEC server, although the data center can provide less execution time, it costs much higher fronthaul and end-to-end delay. To alleviate the competition between edge devices on MEC server resources and take advantage of the powerful computing capability of the data center, offloading requests that accept this delay will be offloaded to the data center.

### *B. Task offloading optimization problem formulation*

The offloading reward of task $i$ to server $j$ is formulated as the profit on delay and energy consumption against local processing. It is written as

$$v_{ij}\left(d_{ij},M_{ij},p_{ij}\right)=d_{ij}\left(\beta_i^{t}\frac{T_i^{l}-T_{ij}^{r}}{T_i^{l}}+\beta_i^{e}\frac{E_i^{l}-E_{ij}^{r}}{E_i^{l}}\right) \tag{8}$$

where $d_{ij}\in\{0,1\}$ denotes the offloading decision of task $i$ (the task will be offloaded when $d_{ij}=1$). $\beta_i^{t}$ and $\beta_i^{e}$ are the preference of task $i$ on latency and energy consumption, where they add up to 1. It is worth noticing that the weight of two terms, $\beta_i^{t}$ and $\beta_i^{e}$, can be interpreted as the preference of individual requests on latency and energy. In practice, they can be determined by applying the multi-attribute utility approach in the multiple criteria decision-making theory [31].

In the network model discussed in the last subsection, every server will receive a random amount of offloading requests at every time slot. The objective of this paper is to maximize the long-term offloading reward of a MEC server network through offloading decision and resource allocation. Therefore, we formulate the problem $P$ as follows:

$$P:\ \max_{\mathcal{D},\mathcal{P},\mathcal{M},\mathcal{G}}\sum_{i=1}^{Q_{jz}}\sum_{j=1}^{U}\sum_{z=1}^{L}V_{ijz} \tag{9a}$$

$$\text{s.t.}\quad C1: Q_{jz}^{d}\le Q_{jz},\qquad \forall j\in U,\ \forall z\in L \tag{9b}$$

$$C2: 0<p_{ij}\le p_0,\qquad \forall i\in Q_{jz}^{d},\ \forall j\in U \tag{9c}$$

$$C3: M_{ij}>0,\qquad \forall i\in Q_{jz}^{d},\ \forall j\in U \tag{9d}$$

$$C4: \textstyle\sum_{i}^{Q_{jz}^{d}}M_{ij}\le G_{jz},\quad \forall j\in U,\ \forall z\in L \tag{9e}$$

$$C5: G_{jz}\le X_z,\qquad \forall j\in U,\ \forall z\in L \tag{9f}$$

where, in 9a, $\mathcal{D}$ denotes the offloading decision policy, $\mathcal{P}$ denotes the transmission power allocation policy, $\mathcal{M}$ represents the computing resource allocation policy between edge devices, and $\mathcal{G}$ represents the long-term accessible resource allocation policy for every group of requests. Overall, the objective function can be interpreted as achieving maximum offloading reward by four joint polices of $\mathcal{D},\mathcal{P},\mathcal{M}$ & $\mathcal{G}$.

Based on Function 8, to take the time dimension into account of the optimization function, $V_{ijz}$ can be written as

$$V_{ijz}=d_{ij}\left(\beta_i^{t}\frac{T_i^{l}-T_{ij}^{r}}{T_i^{l}}+\beta_i^{e}\frac{E_i^{l}-E_{ij}^{r}}{E_i^{l}}\right)/Q_{jz} \tag{10}$$

Within these formulations, $Q_{jz}$ states the number of requests that can not accept the end-to-end delay to the data center but to the server $j$. Taking $Q_{jz}$ as the denominator of Function 10 is to convert the rewards of all the requests within one server into the average rewards of individual users. It is able to eliminate the difference in the number of requests between servers in the MEC network, i.e. the impact of network traffic imbalance. $U$ denotes the collection of servers. $L$ is the time scale for weighing the long-term offloading profits. In constraint $C1$, $Q_{jz}^{d}$ is the collection of the offloaded requests on server $j$ at time slot $z$. Offloaded request number is less than the total amount $Q_{jz}$. Constraint $C2$ states that there is an upper limit on the transmission power for the offloading requests. Moreover, constraint $C3$ ensures all the offloaded tasks will get computing resources. Constraint $C4$ guarantees that the total computing resources distributed to all requests within server $j$ is less than the accessible resources at time slot $z$ determined by $\mathcal{G}$. The last constraint $C5$ explains that the accessible resources to any group of requests at time slot $z$ can not exceed the left resources $X_z$ on their requested servers.

To solve the problem $P$, we decompose it into two subproblems of $P1$ and $P2$.

1)$P1$: Optimization of accessible resource scheduling of each request group over a $U$ servers MEC network for each time slot in $L$.

2)$P2$: Joint optimization of offloading decision and resource allocation in the unit of a single server $j'$ at time slot $z'$.

Accordingly, we can rewrite the problem $P$ as

$$\begin{aligned}P:&\max_{\mathcal{G}}\sum_{j=1}^{U}\sum_{z=1}^{L}\Big(\max_{\mathcal{D},\mathcal{P},\mathcal{M}}\sum_{i=1}^{Q_{jz}}V_{ijz}\Big)\\&\text{s.t.}\quad C1,C2,C3,C4,C5\end{aligned} \tag{11}$$

The long-term profits of $U$ servers determined by $\mathcal{G}$ in range $U$ and $L$ can be separated from problem $P$ as follows

$$\begin{aligned}P1:&F(\mathcal{D},\mathcal{P},\mathcal{M})=\max_{\mathcal{G}}\sum_{j=1}^{U}\sum_{z=1}^{L}K(\mathcal{G})\\&\text{s.t.}\quad C1,C5\end{aligned} \tag{12}$$

where $P2$ can be represented by

$$\begin{aligned}P2:&K(\mathcal{G})=\max_{\mathcal{D},\mathcal{P},\mathcal{M}}\sum_{i=1}^{Q_{jz}}V_{ijz|j=j',z=z'}\\&\text{s.t.}\quad C2,C3,C4\end{aligned} \tag{13}$$

Because of constraint $C4$, $P1$ and $P2$ are interrelated rather than independent of each other. In subproblem $P1$, we explore three DRL algorithms to define $\mathcal{G}$, a temporal dimension

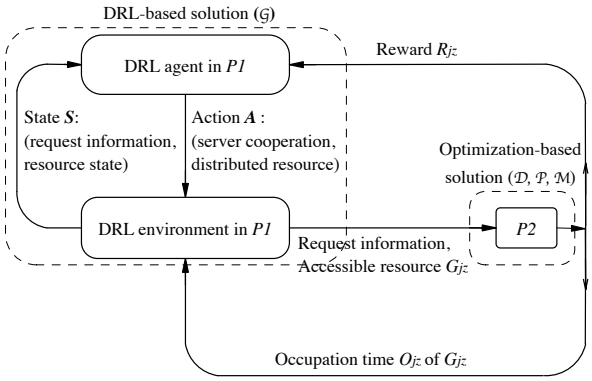



**Fig. 2:** Overall Solution of Task Offloading Management ($P1$: Optimization of long-term resource scheduling; $P2$: Joint optimization of offloading decision and resources allocation).

scheduling policy that reserves the accessible resources $G_{jz}$ for every group of requests. To calculate the required reward and the interaction information between DRL agent and the environment under the limitation of $G_{jz}$, in subproblem $P2$, we apply the optimization-based approach to find the optimal $\mathcal{D}, \mathcal{P}$ and $\mathcal{M}$. Therefore, with these policies, $P2$ can be seen as a function with respect to $G_{jz}$, and its results are taken as the average reward $R_{jz}$ for requests in $Q_{jz}$ and the occupation time $O_{jz}$ of $G_{jz}$. The interactions between $P1$ and $P2$ is shown in Figure 2, and $P2$ can be rewritten as

$$\begin{aligned} P2 :&(R_{jz}, O_{jz}) = \mathcal{D}, \mathcal{P}, \mathcal{M}(G_{jz}) \\ &\text{s.t.} \quad C2, C3, C4 \end{aligned} \tag{14}$$

## III. Integration of DRL Algorithms and Optimization-Based Method

In the MEC network, all the servers incessantly receive offloading requests from their service area. In order to maximize long-term task offloading profit, the MINLP consisting of this objective and other constraints is decomposed into two interrelated subproblems. Subproblem $P1$ is to optimize the accessible resources for each group of requests with $L$ slots over a time span. Taking this as the upper limitation, $P2$ is to make the optimal offloading decision and resource allocation, within each server, between mobile devices. DRL and optimization-based integrated solutions for those two subproblems are discussed in this section.

### A. DRL-based solution of P1

In previous works, the accessible resources for each group of requests were set to all remaining resources on the server at the moment. This policy yields transient high benefits because of the resource shortage of the following requests. Therefore, a long-term polity is needed to decide the accessible resources based on the current resource state and the workload requirement. This temporal dimension accessible resource scheduling policy of $P1$ is set to $\mathbf{G1}$. In addition, we design another scheme to further improve resource utilization. This is when the resources of a server have been run out, the workload can be redirected to other servers via the Mp3 interface instead of being rejected. Moreover, even if a server has sufficient resources, we can still apply this scheme to reduce the execution time by parallel processing. However, workload redirection will complicate the resource management. For instance, when a server lends resources to other servers but receives high demand requests in the next time slot, it can only reject those requests or borrow again from others, and so on to form a vicious circle. Hence, we set another binary policy to decide the workload redirection for the requests. It is set to $\mathbf{G2}$, and $\mathbf{G2} = 1$ when the workload redirection decision is true. To avoid the propagation delay and reduce the complexity of DRL, workload redirection is restricted to happen only between the neighbor MEC nodes. Thus, the overall policy $\mathcal{G}$ can be represented by $(\mathbf{G1}, \mathbf{G2})$. $P1$ can be rewritten as

$$\begin{aligned} P1 :&F(\mathcal{D}, \mathcal{P}, \mathcal{M}) = \max_{\mathbf{G1}, \mathbf{G2}} \sum_{j=1}^{U} \sum_{z=1}^{L} K(\mathbf{G1}, \mathbf{G2}) \\ &\text{s.t.} \quad C1, C5 \end{aligned} \tag{15}$$

The interactions between policies $(\mathbf{G1}, \mathbf{G2})$ and the network is modeled as a MDP [32]. Three DRL algorithms are explored to solve the MDP and find the optimal offloading management policies. In MDP, the discounted reward $R_z$ for the consecutive tasks is defined as

$$R_z = \sum_{k=0}^{\infty} \gamma^k r_{z+k} \tag{16}$$

where $\gamma$ is the discount factor that determines the importance of the transient and future rewards. $k$ is a natural number, and $r_{z+k}$ is the reward at a certain time, $z + k$. Based on Function 16, the action value function is defined to evaluate the action $a_z$ by the discounted return at state $s_z$ as follows:

$$Q_\pi(s_z, a_z) = \mathbb{E}[R_z \mid s_z, a_z] \tag{17}$$

$\pi$ is the policy function and it represents the probability density function of action. Besides, another component, the state value function, is also defined to demonstrate the expected discounted return for selecting state $s_z$ as follows:

$$V_\pi(s_z) = \mathbb{E}[R_z \mid s_z] \tag{18}$$

where $V_\pi$ can be used to assess the quality of the policy function $\pi$. The state value function can also be written as the expectation value of $Q_\pi$ for action set $A$ as follows:

$$V_\pi(s_z) = \mathbb{E}_A[Q_\pi(s_z, A)] = \sum_{a}^{A} \pi(a \mid s_z) \cdot Q_\pi(s_z, a) \tag{19}$$

In general, the essence of our DRL solutions for MDP is to calculate the optimal value function and policy function, which are composed of the reward of now and in the future. Therefore, DRL algorithms are able to learn the offloading policy $\mathcal{G}(G1, G2)$ that balance the long-term benefits and get higher average rewards, i.e. resolving the reward maximization problem formulated in Function 9a and the subproblem $P1$ in Function 12. The next parts of this subsection are concerned with three DRL algorithms.

*1) Single-agent value-based DQN:*

DQN is a novel RL algorithm that combines Q-learning and a deep neural network together to estimate optimal $Q(s,a)$ based on the time differential (TD) algorithm [33]. The basic function of the TD algorithm can be written as

$$Q\left(s_z, a_z\right) \approx r_z + \gamma \cdot Q\left(s_{z+1}, a_{z+1}\right) \tag{20}$$

where $r_z$ is the reward observation at time $z$. $Q\left(s_z, a_z\right)$, $Q\left(s_{z+1}, a_{z+1}\right)$ are the predicted action values of the neural network at time $z$, $z+1$. The right site of equation is closer to the real action value because of actual reward observation $r_z$ and thereby can be set as the TD target. DQN is to minimize the difference between the prediction and the TD target through training. Algorithm 1 shows the detail of the DQN algorithm, where the selection of $a_z$ with probability $\epsilon$ is called greedy policy to avoid the local optimization [33].

To solve subproblem $P1$ with DQN, the offloading process in MEC network consisted of $U$ servers is seen as the MDP. The time slot in the network is linked together with the iterative step shown in Algorithm 3. Since these servers receive requests in every time slot, if $U$ predictions are implemented in one DQN step, overwhelmed state information will cause the action space to explode. Therefore, we implement the management decisions for $U$ groups of requests in one time slot by $U$ consecutive DQN iterations. The offloading reward of the network in each time slot is equal to the average reward of corresponding $U$ iterative steps. The minibatch size of DQN is set to $A$, which represents $A/U$ consecutive time slots.

In the DQN model, the state includes the extracted request information $Q_{jz}$, $\sum_i^{Q_{jz}} C_i$, $\sum_i^{Q_{jz}} S_i$, the number of requests with higher latency preference $\beta_i^t$, and the available computing resources of all servers. Therefore, the size of the state space is $(4+U,)$. The action comprises two objectives, accessible resource allocation, $\mathbf{G1}$ and the workload redirection decision, $\mathbf{G2}$. The action space size is $1*(2*size(\mathbf{G2}))$. Note that the final operation on the network is not only determined by the action but also the remaining resource states. The regulation that actions interact with the network environment is demonstrated in Algorithm 2. For a server j, if $\mathbf{G2}=1$ and its neighbors have remaining resources ($\{X_{neighs}\} \geq \{\sum_i^{Q_{jz}} C_i\}$), it will choose the neighbor with more free resources as its workload redirection destination. In addition, to reduce congestion rates due to $\mathbf{G2}$, in Algorithm 2 we showed that the lending resources from the neighbors will not exceed $\mathbf{G1}/2$ or one third of the remaining resources.

**Algorithm 1** DQN Algorithm

```
Initialize replay memory B
Initialize the neural network with parameter θ
for n in batch do
    Initialise the network resource state
    Get state s_z; select action a_z with probability ε
    Otherwise select a_z = max_a Q*(s_z, a_z | θ_z) [33]
    Execute a_z; get reward r_z and next state s_{z+1}
    Store (s_z, a_z, r_z, s_{z+1}) in B
    Sample minibatch from B
    for m in minibatch do
        Estimate Q(s_m, a_m | θ_m) for s_m
        Estimate Q(s_{m+1}, a_{m+1} | θ_m) for s_{m+1}
        Perform gradient descent based on TD function
    end for
end for
```

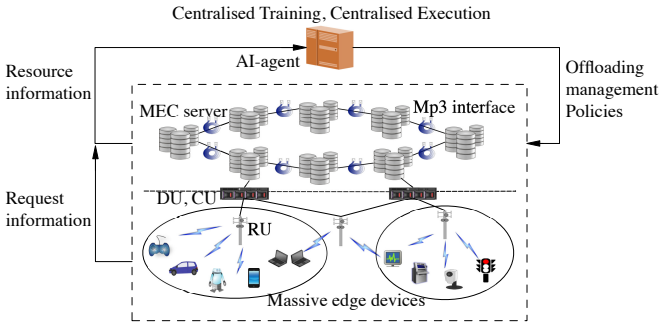



**Fig. 3:** DQN Centralized Training and Centralized Execution Architecture.

Regarding the reward of action $(\mathbf{G1}, \mathbf{G2})$ for a group of requests in $Q_{jz}$, it is calculated based on Function 14. $\mathcal{D}, \mathcal{P}$ and $\mathcal{M}$ of $P2$ will be discussed in the next subsection. Besides, these policies can also help calculate the occupation time of the distributed accessible resources. If a server borrows resources from the others, the occupation time can be interpreted as how soon these resources will be released. The character of subproblem $P2$ in DQN and the basic algorithm structure within one iterative step is demonstrated in Figure 2.

In addition to the algorithm itself, the AI agent placement is also essential for the implementation of DRL technology in the MEC network [34], [35]. In the DQN algorithm, as the AI agent needs to make policies for all servers, it needs to be located on a centralized controller which is aware of all the requests and available resource information in real-time. Because both offline training and online execution take place in the controller, this is a centralized training and centralized execution architecture. The implementation of DQN in the real network is shown in Figure 3.

Based on Function 20, since the management of $U$ servers in one slot is realized by $U$ iterations, the relationship among $U$ servers can be seen as cooperating to obtain the maximum average resources. However, the discount factor $\gamma$ in the action value function will weaken the cooperation between servers. In addition, DQN divides the sharing of resources between servers in a one-time slot into $U$ iterations. This might reduce the effectiveness of the learned policies $\mathbf{G1}$ and $\mathbf{G2}$ on long-term benefits balancing. Moreover, since the centralized controller is placed beyond the server, in order to ensure the reliability of the service in the real network, the upload of request and resource information and the download of management decisions have strict requirements on deterministic communication [36]. In this vein, if the network can not ensure real-time traffic, DQN might not be applicable to solve this problem.

*2) Multi-agent policy-based DDPG:*

In addition to DQN, multi-agent policy-based DDPG is also used to solve $P1$ and find the optimal $(\mathbf{G1}, \mathbf{G2})$. MADDPG

**Algorithm 2** Interaction regulation between agent and network

Processing stage:
**if** $\mathbf{G2} = 1$ **then**
  $G_{jz1} = \min\{\mathbf{G1}, X_j\}$
  $G_{jz2} = \min\{\mathbf{G1}/2, \max\{X_{neighs}/3\}\}$
  $G_{jz} = G_{jz1} + G_{jz2}$
**else**
  $G_{jz} = \min\{\mathbf{G1}, X_j\}$
**end if**

The role of $X_j$ in Co-decision:
**if** $X_j < \sum_i^{Q_{jz}^d} C_i$ & $\{X_{neighs}\} < \{\sum_i^{Q_{jz}^d} C_i\}$ **then**
  No offloading
**else if** $X_j \geq \sum_i^{Q_{jz}^d} C_i$ & $\{X_{neighs}\} < \{\sum_i^{Q_{jz}^d} C_i\}$ **then**
  No workload redirection
**else if** $X_j < \sum_i^{Q_{jz}^d} C_i$ & $\{X_{neighs}\} \geq \{\sum_i^{Q_{jz}^d} C_i\}$ **then**
  Achievable workload redirection
**else if** $X_j \geq \sum_i^{Q_{jz}^d} C_i$ & $\{X_{neighs}\} \geq \{\sum_i^{Q_{jz}^d} C_i\}$ **then**
  Achievable workload redirection
**end if**

is another promising RL algorithm that uses policy gradient to estimate the maximum state value $V_\pi(s_z)$ [37]. Because the state value function is equal to the expectation value of $Q_\pi$ as shown in Function (19), the main idea of DDPG is to get $V_\pi(s_z)$ by using two neural networks to approximate the policy function $\pi(a \mid s_z)$ and the action-value function $Q_\pi(s_z, a)$. This is realised by the actor-critic architecture [38]. Within this architecture, the actor neural network applies the policy gradient algorithm to optimize the policy function; the critic network uses the same training method with DQN to estimate the corresponding action-value function. The introduction of MADDPG facilitates the applications that need to consider the interaction between multi-agent. Moreover, this system effectively solves the action space explosion by decomposing the multi objectives to different agents.

In MADDPG, each agent needs a group of actor, target actor, critic and target critic neural networks. Each actor is trained with independent network observation (Obs $S_j$). Furthermore, to enable interactions between agents, the actions decided by all actors are shared as input among each critic. The parameters of target networks are updated from corresponding actor and critic networks based on

$$P_t = \tau P + (1 - \tau) P_t \tag{21}$$

where $\tau$ is the target network update coefficient. $P$ and $P_t$ are the parameter of evaluation networks and target networks, respectively [38]. It is designed to increase the converging speed. The multi agent actor-critic structure is shown in Figure 4.

To define the optimal accessible resource allocation and workload redirection decision, we apply MADDPG in the same network environment as DQN. Within this technology, each pair of actor and critic corresponds to a server. Moreover, based on the actor-critic structure, the interaction between servers on the limited resources in one time slot is achieved

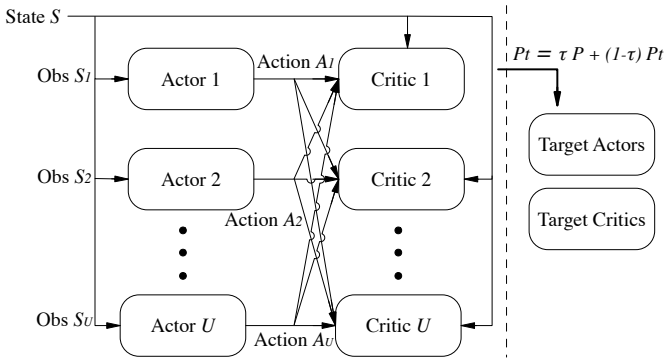



**Fig. 4:** The Structure of DRL Multi-Agent Actor-Critic.

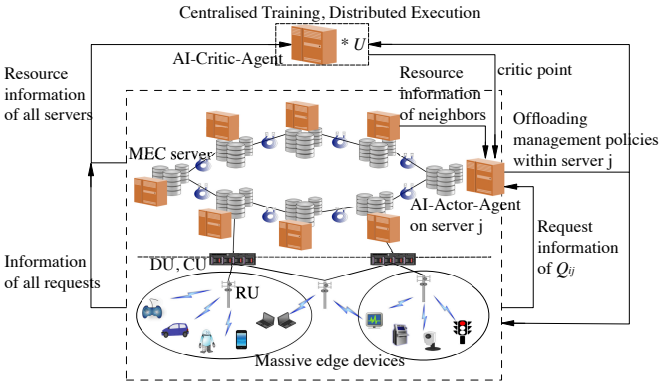



**Fig. 5:** MADDPG Centralized Training and Distributed Execution Architecture.

though a single iteration step.

In each actor, the action is consistent with DQN, including the workload redirection decision and the accessible resource allocation. Because the action in DDPG is Different from DQN, the state comprises the extracted request information and the available computing resources of the processing server and its neighbors. The resource information of the entire network is not required [39]. The state space size is $1 * (4 + 1 + Neighbors)$. In each critic, the output with shape $1 * 1$ is used as the criterion of the policy gradient algorithm for the corresponding actor, and the input includes not only the request and resource information of the entire network, but also the actions of other actors. Its shape is $1 * (4 * U + 2 * U)$.

Similarly to the architecture shown in Figure 2, the reward of an action is also calculated by solving $P2$. However, we designed two different methods to set the reward for the critic neural networks, which represent two interaction forms between the servers. The first is fully cooperative (FC) method. FC gets the rewards of $U$ actors and takes their average as the common reward for all critics. It is designed to let all the servers take into account the limited resources and make reasonable workload redirection decisions to obtain higher offloading benefits. The other is self-interested (SI) method. It applies individual rewards from the actors to their corresponding critics, which means all the critics update their respective neural networks with different references. Within this method, each server wants to get the best reward even if their actions might impair the interests of

**Algorithm 3** MADDPG Algorithm

Initialize replay memory $\mathcal{B}$
Initialize the actor, target actor, critic and target critic with parameter $\theta_{1\sim U}, \theta^*_{1\sim U}, \omega_{1\sim U}, \omega^*_{1\sim U}$
**for** $n$ in batch **do**
  Initialise the network resource state
  Get state $s_{z,1\sim U}$
  Estimate $a_{z,1\sim U}$ by $\pi(s_{z,1\sim U}; \theta_{z,1\sim U})$
  Execute $a_{z,1\sim U}$, get reward $r_{z,1\sim U}$ and next state $s_{z+1}$
  Store $(s_{z,1\sim U}, a_{z,1\sim U}, r_{z,1\sim U}, s_{z+1,1\sim U})$ in $\mathcal{B}$
  Get minibatch from $\mathcal{B}$
  **for** $m$ in minibatch **do**
    Estimate $a_{m+1,1\sim U}$ by $\pi(s_{m+1}; \theta^*_{m,1\sim U})$
    Estimate $Q(s_{m,1\sim U}, a_{m,1\sim U} \mid \omega_{1\sim U})$
    Estimate $Q(s_{m+1,1\sim U}, a_{m+1,1\sim 8} \mid \omega^*_{1\sim U})$
    **if** Full cooperation method **then**
      Use common reward for critics, $r_z = \overline{r_{z,1\sim U}}$
    **else if** Self interested method **then**
      Use individual reward for critics, $r_{z,1\sim U}$
    **end if**
    Perform gradient descent for critic based on TD function
    Get $d_{\omega,\theta,t} = \frac{\partial Q(s,\pi(s;\theta);\omega)}{\partial \theta}|_{\theta=\theta_{m,1\sim U},\omega=\omega_{m,1\sim U}}$
    Perform gradient ascent for actor based on policy gradient algorithm:
    $\theta_{m+1,1\sim U} = \theta_{m,1\sim U} + \mu d_{\theta_{m,1\sim U},\omega_{m,1\sim U}}$
    Update target network parameter based on
    $\theta^*_{1\sim U} = \tau * \theta_{1\sim U} + (1-\tau) * \theta^*_{1\sim U}$
    $\omega^*_{1\sim U} = \tau * \omega_{1\sim U} + (1-\tau) * \omega^*_{1\sim U}$
  **end for**
**end for**

others, thereby, causing The Tragedy of the Commons [40]. To avoid this situaction, we set constraint to restrict actions that overuse resources. Both of these methods are able to solve subproblem $P1$ and increase the long-term resource utilization. The algorithms of SI-MADDPG and FC-MADDPG are demonstrated in the Algorithm 4, where $\tau$ in the policy gradient algorithm is the coefficient to update the target network parameter. $\theta$, $\theta^*$, $\omega$, $\omega$ are parameters of actor, target actor, critic, and target critic, respectively. The minibatch size is set to $A/U$ corresponding to the $A$ of DQN.

As mentioned earlier, in order to apply MADDPG to solve $P1$, each pair of actor and critic needs to know the extracted request information and the available resources of this server and its neighbors. In addition, critics also need to be aware of the actions of all actors so as to learn the best workload redirection decision, $\mathbf{G2}$. Therefore, in the real network, we place one actor agent on every server, and $U$ critic agents together on a centralized controller. Because actors can make the actions without the knowledge of critics after training, this is a centralized training and distributed execution architecture. The implementation of MADDPG in the real network is shown in Figure 5.

With multi agents, MADDPG overcomes the shortcoming of DQN that only allows interaction between servers to be realized by separating one time slot into $U$ iterative steps. Because the offloading management decisions are shared between all servers to learn the best overall policies, MADDPG may achieve better offloading policy, $\mathbf{G2}$ with higher resource utilization. In addition, actors are located on servers and they only need to know the available computing resources of their neighbors, so that each actor can realize its own action without communication with the centralized controller. It highly reduces the execution delay and thus has less latency requirement on the network.

### *B. Optimization-based DCTRA of P2*

With the result of $P1$ on accessible resource $G_{jz}$ for each request group in the MEC network, as discussed earlier, $P2$ is to design a joint offloading management policy, including $\mathcal{D}$, $\mathcal{P}$, & $\mathcal{M}$, to maximize the utilization of $G_{jz}$ and the offloading profits of the requests within server $j$. The results will be input back to $P1$ as the DRL reward and interaction information between agent and environment.

Problem $P2$ has been proven to be an NP hard problem in [13]. According to the Tammer decomposition method [41], subproblem $P2$ can be further decomposed into two independent sub-sub problems of lower complexity. Their constraints on the $\mathcal{D}, \mathcal{P}$ and $\mathcal{M}$ are able to be decoupled from each other. Sub-sub problem $P3$ can be rewritten as

$$P3: J(\mathcal{P}, \mathcal{M}) = \max_{\mathcal{D}} H(\mathcal{D}) \tag{22}$$

in which $P4$ can be denoted as

$$\begin{aligned} P4: & H(\mathcal{D}) = \max_{\mathcal{P},\mathcal{M}} \sum_{i}^{Q_{jz}} V_{ijz|j=j',z=z'} \\ & \text{s.t. } C2, C3, C4 \end{aligned} \tag{23}$$

We solve these two sub-sub problems in the order of $\mathcal{P}, \mathcal{M}$ & $\mathcal{D}$.

*1) Optimization of transmission power allocation $\mathcal{P}$:*

Substituting (8) into (23), we can rewrite the objective function of $P4$ as

$$\max_{\mathcal{D},\mathcal{P}} \Big[ \sum_{i}^{Q_{j'z'}} \left(\beta_i^t + \beta_i^e\right) - \sum_{i}^{Q_{j'z'}} \left(\beta_i^t T_{ij'}^r / T_i^l + \beta_i^e E_{ij'}^r / E_i^l\right) \Big] \tag{24}$$

Because $\beta_i^t + \beta_i^e = 1$, $P4$ turns into finding the minimum results of the second term. Bringing all the network information discussed in the system model into (23), it can be rewritten as

$$\begin{aligned} P4: & \min_{\mathcal{P},\mathcal{M}} \sum_{i}^{Q_{j'z'}} \left(\frac{(\eta_i + \gamma_i p_{ij'})}{\log_2(1 + a_{ij'} p_{ij'})} + \frac{\beta_i^t T_{ij'}^e}{T_i^l}\right) \\ & \text{s.t. } C2, C3, C4 \end{aligned} \tag{25}$$

where $\eta_i = \beta_i^t S_i / W T_i^l$ and $\gamma_i = \beta_i^e S_i / W E_i^l \zeta$. The first term of the objective in (25) can be used to calculate the transmission power allocation policy as

$$\begin{aligned} & \min_{\mathcal{P}} f(p_{ij'}) = \frac{\eta_i + \gamma_i p_{ij'}}{\log_2(1 + a_{ij'} p_{ij'})} \\ & \text{s.t. } C2 \end{aligned} \tag{26}$$

**Algorithm 4** Transmission Power Allocation Policy $\mathcal{P}$

Set a small number $\delta$
Calculate $\phi(p_0) = \gamma_i \log_2(1 + a_{ij'}p_0) - a_{ij'}/\ln 2 \cdot (\eta_i + \gamma_i p_0)(1 + a_{ij'}p_0)$
**for** $i$ in $Q^d_{j'z'}$ **do**
  **if** $\phi(p_0) = 0$ **then**
    $p_{ij'} = p_0$
  **else**
    Initialize $p_s = 0$ and $p_t = p_0$
    **while** $p_t - p_s \leq \delta$ **do**
      $p_l = (p_t + p_s)/2$
      **if** $\phi(p_l) \leq 0$ **then**
        $p_s = p_l$
      **else**
        $p_t = p_l$
      **end if**
      $p_{ij'} = (p_t + p_s)/2$
    **end while**
  **end if**
**end for**

**Algorithm 5** Offloading Decision Policy $\mathcal{D}$

Based on delay requirement, select $Q_{jz}$ from $Q_z$
Offload $Q_z - Q_{jz}$ to data center
$time = 0$
**for** $i$ in $Q_{jz}$ **do**
  $time = time + C_i/G_{jz}$
  **if** $\beta^t_i > \beta^e_i$ & $time \geq T^l_i$ & $V_{ij'z'} < 0$ **then**
    Reject
  **else if** $\beta^e_i > \beta^t_i$ & $E^r_{ij'} \geq E^l_i$ & $V_{ij'z'} < 0$ **then**
    Reject
  **else if** $V_{ij'z'} < 0$ **then**
    Reject
  **else**
    Offload to $j'$
  **end if**
**end for**

The first order derivative of $f(p_{ij'})$ is

$$f'(p_{ij'}) = \frac{\gamma_i \log_2(1 + a_{ij'}p_{ij'}) - a_{ij'}/\ln 2 \frac{\eta_i + \gamma_i p_{ij'}}{1 + a_{ij'}p_{ij'}}}{\log^2_2(1 + a_{ij'}p_{ij'})} \quad (27)$$

$f'(p_{ij'}) = 0$ , when

$$\phi(p_{ij'}) = \gamma_i \log_2(1 + a_{ij'}p_{ij'}) - a_{ij'}/\ln 2 \cdot \frac{\eta_i + \gamma_i p_{ij'}}{1 + a_{ij'}p_{ij'}} = 0$$

and $\phi(0) = -a_{ij'}\eta_i/\ln 2 < 0$.

The second order derivative of $f(p_{ij'})$ is

$$f''(p_{ij'}) = \frac{a^3_{ij'}}{\gamma_i \ln^2 2} \frac{(\eta_i + \gamma_i p_{ij'})^2}{(1 + a_{ij'}p_{ij'})^3 \log^3_2(1 + a_{ij'}p_{ij'})} \quad (28)$$

It is easy to prove that $f''(p_{ij'}) \geq 0$. Therefore, $f(p_{ij'})$ is a quasi-convex function in the domain.

Overall, with the property of the convex function, we designed a bisection policy to optimize the transmission power allocation, shown in Algorithm 4.

*2) Optimization of offloading decision $\mathcal{D}$ and computing resource allocation $\mathcal{M}$:*

The second term of the objective of $P4$ can be rewritten into

$$w(\mathcal{M}) = \min_{\mathcal{M}} \sum_i^{Q_{j'z'}} \frac{\beta^t_i T^e_{ij'}}{T^l_i} \quad \text{s.t. } C3, C4 \quad (29)$$

where

$$\sum_i^{Q_{j'z'}} \frac{\beta^t_i T^e_{ij'}}{T^l_i} = \sum_i^{Q_{j'z'}} \frac{\beta^t_i f^l_i}{G_{j'z'}} \quad (30)$$

If the policy $\mathcal{M}$ is that all the offloaded tasks share the computing resources $G_{jz}$ simultaneously and end at the same time, $w(\mathcal{M})$ can be calculated as

$$w(\mathcal{M})_{syn} = \sum_i^{Q^d_{j'z'}} \frac{\beta^t_i f^l_i C_i}{G_{j'z'} Q^d_{j'z'}} \quad (31)$$

where the set of offloaded requests $Q^d_{jz}$ is determined by $\mathcal{D}$, and $d_{ij}$ of all the requests in $Q^d_{jz}$ are equal to 1. In contrast, HODA [13] applied Hessian matrix and Lagrangian dual function in (29) and obtained the value of $w(\mathcal{M})$ as

$$w(\mathcal{M})_{HODA} = \frac{\left(\sum_i^{Q^d_{j'z'}} \sqrt{\beta^t_i f^l_i}\right)^2}{G_{j'z'}} \quad (32)$$

By the inequality of arithmetic and geometric means [42], it is easy to prove that

$$\frac{\left(\sum_i^{Q^d_{j'z'}} \sqrt{\beta^t_i f^l_i}\right)^2}{G_{j'z'}} \leq \frac{Q^d_{j'z'} \sum_i^{Q^d_{j'z'}} \beta^t_i f^l_i}{G_{j'z'}} \quad (33)$$

In the situation when equality is established in (33), by comparing (31) and the right term of (33), we can get that when $(Q^d_{j'z'})^2 > \sum_i^{Q^d_{j'z'}} C_i$, simultaneously processing solution performs better than HODA.

In addition, we propose to sort the tasks for executions according to $\beta^t_i/C^l_i$ from large to small. The task with less CPU requirement but high latency preference will be prioritized. Note that the resources released by the finished task will be reallocated to the next task and so on. The $w(\mathcal{M})$ produced in this method can be written as

$$w(\mathcal{M})_{sorted} = \sum_i^{Q^d_{j'z'}} \frac{(Q^d_{j'z'} - i + 1)\beta^t_i f^l_i}{G_{j'z'} Q^d_{j'z'}} \quad (34)$$

where the waiting time is considered. It is easy to prove that $w(\mathcal{M})_{sorted} \ll w(\mathcal{M})_{HODA}$.

To calculate $Q^d_{jz}$ in Functions 31 to 34 and Algorithm 4, we designed the offloading policy $\mathcal{D}$ as shown in Algorithm 5. The first step of Algorithm 5 is to filter out the request that going to the data center.

Overall, when a certain server $j'$ receives some requests $Q_{j'z'}$ at time slot $z'$, with $\mathcal{P}, \mathcal{M}$ and $\mathcal{D}$, we are able to calculate the reward $R_{j'z'}$ and the occupation time $O_{j'z'}$ of the accessible resources $G_{j'z'}$ as shown in Function 14. These results are able to be further input into the DRL algorithms to learn the policy $\mathcal{G}$.

## IV. Simulations and Results

In this section, we first provide the setup information of MEC servers and mobile devices in the edge computing network and the summary of neural networks of three DRL algorithms. Based on the setup, we study the offloading benefits of DCTRA and several other algorithms under different accessible resources in one time step. In the end, to demonstrate the advantage of DRL in long-term resource planning and the necessity of server cooperation, the reward distribution of several scheduling policies over a period of time are also explored in this section.

### A. Experimental Setup

The experimental scenario is set as follows:

- In the MEC network, there are 8 servers with the same amount of computing resources, 100 Intel i7-1195G7 CPU modules with maximum overall 500GHz processing capability. These servers are in the ring topology so that each server is connected to two others. To reflect the unbalanced traffic property of the practical network, the number of mobile devices served by each server was randomly set between 700 and 1000. In each time slot, 20% to 30% of these mobile devices will send the offloading requests to the servers, and half of them are not able to tolerate the delay to the data center. The other properties of the network include the maximum transmit power $p_0$, 20dBm and the noise density, -174dBm/Hz.
- For mobile devices, we assume that they have similar local processing capabilities from 1GHz to 3GHz. The user bandwidth $W$ is set to 3MHz for calculating the data rate. In addition, their preference on delay and energy consumption $\beta_{ij}^T$ and $\beta_{ij}^E$ varies between 1/6 and 5/6. The offloading requests sent by mobile devices contain the required data size $S_i$, 10 to 20 in the unit of MB, and the required amount of computing resources, 10 to 20 in the unit of Gcycles.
- In the DRL part, we use 50 time slots as $L$ to train the model. Therefore, the minibatch size of DQN, $A$, is set to 400 and the corresponding minibatch size of two MADDPG algorithms is set to 50. The action space of $\mathbf{G2}$ is distributed between 50GHz to 150GHz. To compare the training cost, the learning rate $\mu$ of both DQN and MADDPG is set to $10^{-5}$. The greedy policy probability is set to 0.2. The target network update coefficient $\tau$ of MADDPG and the reward discount $\gamma$ of all three DRLs are set to 0.1, and 0.99, respectively. The neural network components of DQN and MADDPG algorithms is summarized in Table II.

### B. Comparison of Joint offloading decision and resource allocation algorithms

Figure 6 compares the average offloading rewards of several management policies for 150 offloading requests and their variations with different available resources. These rewards come from the result of the $P2$. In Figure 6, black, red, blue, purple, and green lines represent the result of DCTRA, fully offloading (FO), HODA, distributed game theory offloading (DGTO) and local processing (LP), respectively. Besides our proposed DCTRA, LP is defined to process all the requests on mobile devices. FO is to offload all the requests to MEC servers without offloading decisions. The reallocation of resources is also considered in FO. HODA is a semi-distributed approach which is proposed in [13] to ensure mobile users with better utility can be preferentially offloaded. DGTO in [15] is a distributed computation offloading algorithm that can achieve a Nash equilibrium through a multi-user computation offloading game to offload tasks from the benefited edge devices. The Nash equilibrium in the network can be written as

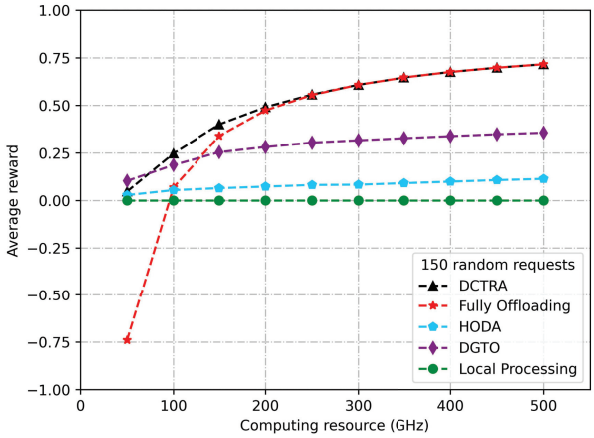



**Fig. 6:** Reward Comparison Between Different Joint Offloading Local Algorithms in $P2$. DCTRA is our designed optimization-based joint offloading management policy. DGTO and HODA stand for distributed game theory offloading and heuristic offloading decision proposed in [15] and [13], respectively.

$$V_{ijz}\left(d_{ij}, d^*_{-ij}\right) \geq V_{ijz}\left(d^*_{ij}, d^*_{-ij}\right), \forall i \in Q_{jz}, \forall j \in U, \forall z \in L \tag{35}$$

where $d^*_{ij}$ denotes the strategy of the edge devices $i$ connect to $j$ and $d^*_{-ij}$ denotes the $Q_{jz}-1$ strategies of the rest edge devices except $i$. It ensures no request can further reduce its offloading rewards by unilaterally changing its strategy and every offloaded requests can get benefits from edge computing approach. Among them, the green line remains at zero because the objective function 13 is formulated as the profit against LP. In addition, the red line in Figure 6 shows that FO algorithm could only get negative rewards when the resources were less than 97GHz. It explains that blindly offloading can severely hinder the execution speed when the resources are insufficient. In comparison, DCTRA with offloading decision and resource allocation greatly improves the utilization of limited resources. It can be seen that when the resource is 100GHz, DCTRA gets approximately 5 times higher rewards than FO and HODA. After computing resources exceed 250GHz, sufficient resources allow DCTRA to offload all the tasks to the server-side, causing the black and red lines to coincide.

Furthermore, in Figure 6, it is apparent that DGTO outperforms all other solutions when the resource is very limited. The main reason is that DCTRA and HODA do not explore more offloading possibilities, but use sorting with first-fit based solutions for edge devices. In contrast, DGTO allows assigning

the fixed resources to fewer offloading tasks that lead to higher rewards. However, when the resources exceed 75GHz, DCTRA has a significant advance in offloading rewards compared to HODA and DGTO. This discrepancy could be attributed to execution ordering and re-utilization of released resources within the same set of requests. HODA and DGTO do not consider the occupation time and the reallocation of resources released from tasks finished in advance. In contrast, DCTRA dictates both the volume and the occupation time of resources for a set of requests. During the occupation time, released resources will be reallocated to subsequent requests, thereby greatly increasing the rewards.

### C. Complexity and cost of DRL algorithms

The computational complexity of DRL algorithms is important to determine their feasibility in practical networks. Thus, we analyze both the time and space complexity and execution cost of single-agent DQN and MADDPG in this subsection.

Refering to [43], the time complexity of reinforcement learning is sublinear in the length of the state period, and the space complexity is sublinear in the size of state space, action space, and step numbers per episode. Therefore, the time complexity of DRL can be represented by $O$(training steps). We summarize the convergence property of DQN, SI-MADPPG, and FC-MADDPG in Figure 7. Of these three algorithms, SI-MADDPG and FC-MADDPG require similar training steps to converge. In contrast, more training steps are required for the DQN algorithm, which separates the time slot into multiple iterative steps. Besides, the space complexity can be written as $O(N_s N_a N_h U)$, where $N_s$ is the number of states, $N_a$ is the number of actions, and $N_h$ is the number of training steps in each episode. Based on these values mentioned in Section III and Subsection IV-A, the space complexities of DQN and MADDPG for our scenario can be calculated as $O(96000)$ and $O(62400)$, respectively. MADDPG can achieve lower space complexity as it reduces the action and state space of DQN into smaller pieces.

In addition, we summarize the shape and components of neural networks of DQN and MADDPG in Table II. Since DRL algorithms have different neural network architectures and activation functions that affect the feature of back propagation, to verify the cost of implementing these algorithms in the real network, their execution time is also provided. Based on the machine with EVGA GeForce GTX 1080 (Base Clock: 1708 MHZ; CUDA Cores: 2560), the training times of DQN and MADDPG are around 16 hours and 40 hours, respectively. Although MADDPG has lower time and space complexity, it takes a longer training time as it requires 7 more runs of the environment and reward calculation per training step than single-agent DQNs. Another cost of MADDPG is that it require distributed agent on every MEC server, which might increase the overhead of neural network update and management.

### D. Comparison of temporal dimension accessible resource allocation algorithms

To prove the effectiveness of DRL and server cooperation, besides DQN and two MADDPG algorithms discussed earlier, random allocation (RA), over-allocation (OA), and FC-MADDPG without server cooperation function (FC-MADDPG-NSC) are also simulated in the MEC network. Within them, RA is to randomly select $\mathbf{G1}$ and $\mathbf{G2}$ in the action space. OA is an over-allocation technique used in previous research with the addition of server cooperation, which fixes $\mathbf{G1}$ and $\mathbf{G2}$ to 1 and 150GHz.

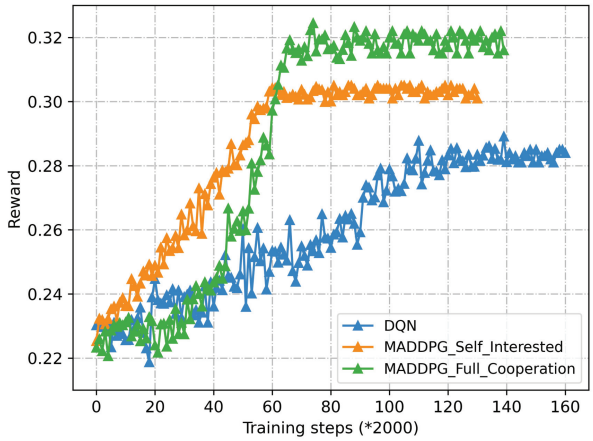


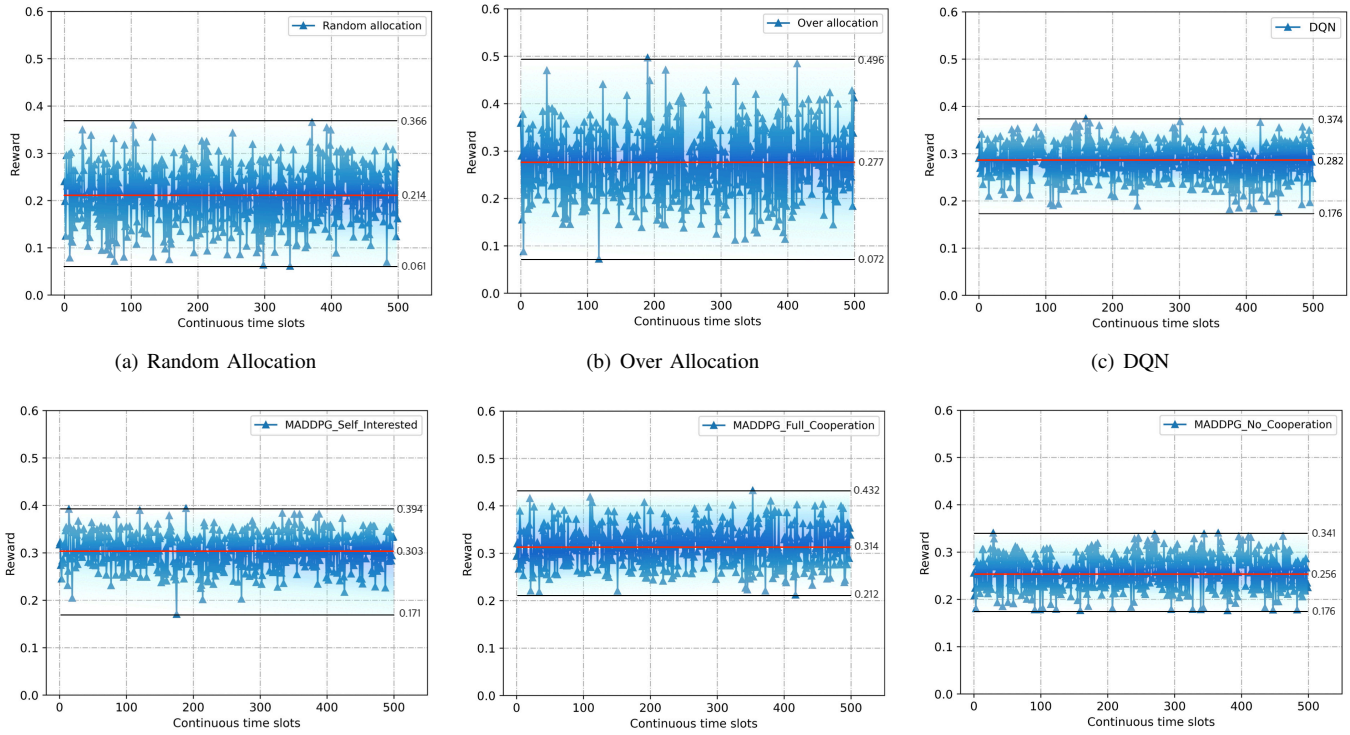

**Fig. 7:** The Convergence Property of Resource Scheduling DRL Algorithms in $P1$.

**TABLE II:** The Summary of Neural Networks

| | Layer | Shape | Activation |
|---|---|---|---|
| **DQN** | Input | 4 (Request info) + 8 (All server resource) | - |
| | Fully connected | 64 | ReLU |
| | Fully connected | 32 | ReLU |
| | Output | 2 (G1) * 10 (G2) | ReLU |
| **MADDPG** *Agent* | Input | 4 (Request info) + 3 (Neighbor server resource) | - |
| | Fully connected | 32 | ReLU |
| | Fully connected | 16 | ReLU |
| | Output | 1 (G1) + 1 (G2) | Tanh |
| *Critic* | Input | 8 (All actor action) + 8 (All server resource) + All request information (4 * 8) | - |
| | Fully connected | 64 | ReLU |
| | Fully connected | 32 | ReLU |
| | Output | 1 | ReLU |

Each figure in 8(a)-8(f) demonstrates the average reward for 8*500 groups of requests that are connected to 8 servers in 500 continuous-time slots. They use the unified algorithm (DCTRA) to maximize the allocated resource and calculate the reward and occupation time for every group of requests, whereas they apply different policies to determine how much resources each group of requests can get access to. 500 continuous-time slots are selected to demonstrate their long-term profits. 8(a), 8(b), 8(c), 8(d), 8(e), 8(f) represent RA, OA, DQN, SI-MADDPG, FC-MADDPG, and FC-MADDPG-NSC, respectively. The main characteristics of the reward of these six resource allocation algorithms are illustrated in Table III.

RA has the worst performance. It has a 43.4% reward distribution between 0 to 0.2, and its rewards over 500 time slots are on average 0.214 and the lowest out of the five algorithms, even worse than FC-MADDPG-NSC without server cooperation. In addition, the method used in previous research,




(a) Random Allocation (b) Over Allocation (c) DQN

(d) SI-MADDPG (e) FC-MADDPG (f) FC-MADDPG-NSC

**Fig. 8:** Reward Distribution of Accessible Resource Scheduling Algorithms in $P1$ over 500 Continuous Time Slots.

**TABLE III:** Summary of Figure 8

| | Random Allocation | Over Allocation | DQN | SI-MADDPG | FC-MADDPG | FC-MADDPG-NSC |
|---|---|---|---|---|---|---|
| Max Reward | 0.366 | 0.496 | 0.374 | 0.394 | 0.432 | 0.341 |
| Min Reward | 0.061 | 0.072 | 0.176 | 0.171 | 0.212 | 0.176 |
| Average Reward | 0.214 | 0.277 | 0.282 | 0.303 | 0.314 | 0.256 |
| Variance | 3.82E-03 | 5.53E-03 | 1.69E-03 | 1.32E-03 | 1.63E-03 | 1.35E-03 |
| Range 0 - 0.1 | 4% | 0.4% | 0% | 0% | 0% | 0% |
| Range 0.1 - 0.2 | 39.4% | 15.8% | 2.4% | 0.2% | 0% | 5.8% |
| Range 0.2 - 0.3 | 49.2% | 46.6% | 60.6% | 44.8% | 64.2% | 81% |
| Range 0.3 - | 7.4% | 37.2% | 37% | 55% | 35.8% | 13.2% |

OA, can achieve the maximum reward of 0.496, whereas it has 16.2% distribution in the range of 0 to 0.2. This might be because OA will yield high short-term gains at the cost of long-term losses. Its highest variance can also be evidenced for this reason.

Compared to those two unreliable solutions, DQN, SI-MADDPG, and FC-MADDPG are three balanced planning solutions that aim for long-term rewards maximization. As shown in Figure 8 and Table III, these algorithms have more stable and higher average rewards than the first two solutions. Among them, although MADDPG and DQN have the same property of server cooperation, the advantage of FC-MADDPG over DQN in long-term average reward could be identified from Table III. This gap may come from the inaccuracy of the cooperation equivalent method of DQN, which separates the cooperation management in a one-time slot into 8 iterative DRL steps. Moreover, during the training process, the agent in MADDPG will also take the actions of other agents as cooperation references. Although MADDPG has better performance than single-agent DQN, the achieved improvement is at the expense of execution time. In addition, FC-MADDPG has a slight improvement over SI-MADDPG. A possible explanation is that the self-interested learning mode strives to maximize its own interests, which may damage the overall rewards.

Figure 8(f) shows the reward when FC-MADDPG is applied in the network without workload redirection **G2**. Compared to 7(e), there is a severe decline in the reward. In FC-MADDPG-NSC, since the workload cannot be processed in parallel with the help of neighbor node resources, it is difficult for FC-MADDPG-NSC to obtain higher rewards than other DRLs. Its average return is the lowest among these algorithms except for random allocation. It proves the importance of a multi-server cooperation in improving resource utilization compared to the single server system.

In these results, the size of the time slot indicates the responding speed of the system. As it increases, more requests will be received on the server within one slot, resulting in

longer average execution delays and lower average rewards. Nevertheless, long-term scheduling may still benefit from DRL compared to other algorithms. Future studies could examine the benefits of DRL in edge networks with different latency requirements. In addition, the power consumption of DU/CU modules has not been studied in-depth in this paper. Furthermore, although the placement of the AI agent in the real network is discussed in the paper, the overhead of training on energy and computing resources remains unclear and requires further exploration in future work.

## V. Conclusion

This paper has argued that the emerging 5G computing-intensive, latency-sensitive applications exceed the bearing capacity of the MEC server. Overloaded MEC servers without effective offloading management policies may get worse results than local execution. In this paper, we build a system where mobile devices have wireless access to multiple servers and can be further redirected to neighbor servers through wired connections. The delay and energy consumption relative to local execution is taken as the offloading benefit criteria. With the purpose of maximizing the long-term reward, we develop a joint management policy with future planning properties through the integration of the optimization-based method and three different DRL algorithms. Simulation results show that DRL can gain great improvements compared to previous methods that only consider the instant return. In addition, with the expense of complexity, multi-agent DRL can more precisely take account of the interaction between servers and thereby realizing higher average offloading benefits than single-agent DQN.

## Acknowledgments

The authors like to thank the support from China Scholarship Council and European Commission's Horizon 2020 research and innovation program under grant agreement No 871428, 5G-CLARITY project. In addition, this study is also financed in part by the Alan Turing Institute - UK (project code: D-ACAD-052) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

## References

[1] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. Rodrigues, and M. Guizani, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," IEEE communications magazine, vol. 56, no. 2, pp. 44–51, 2018.

[2] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, S. Garg, D. Puthal, P. James, A. Zomaya et al., "Iotsim-edge: a simulation framework for modeling the behavior of internet of things and edge computing environments," Software: Practice and Experience, vol. 50, no. 6, pp. 844–867, 2020.

[3] M. Diamanti, P. Charatsaris, E. E. Tsiropoulou, and S. Papavassiliou, "Incentive mechanism and resource allocation for edge-fog networks driven by multi-dimensional contract and game theories," IEEE Open Journal of the Communications Society, vol. 3, pp. 435–452, 2022.

[4] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5g mobile edge computing: architectures, applications, and technical aspects," IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 1160–1192, 2021.

[5] S. S. Gill, P. Garraghan, V. Stankovski, G. Casale, R. K. Thulasiram, S. K. Ghosh, K. Ramamohanarao, and R. Buyya, "Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge," Journal of Systems and Software, vol. 155, pp. 104–129, 2019.

[6] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," IEEE Transactions on Wireless Communications, vol. 17, no. 3, pp. 1784–1797, 2017.

[7] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A drl agent for jointly optimizing computation offloading and resource allocation in mec," IEEE Internet of Things Journal, vol. 8, no. 24, pp. 17 508–17 524, 2021.

[8] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," IEEE/ACM Transactions on Networking, vol. 28, no. 3, pp. 1405–1418, 2020.

[9] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in uav-assisted multi-access edge computing systems under resource uncertainty," IEEE Transactions on Mobile Computing, 2021.

[10] S. Li, Y. Liu, X. Qin, Z. Zhang, and H. Li, "Task offloading and resource allocation in heterogeneous edge computing systems," in 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). IEEE, 2021, pp. 1–6.

[11] J. Xue and Y. An, "Joint task offloading and resource allocation for multi-task multi-server noma-mec networks," IEEE Access, vol. 9, pp. 16 152–16 163, 2021.

[12] M. Nduwayezu, Q.-V. Pham, and W.-J. Hwang, "Online computation offloading in noma-based multi-access edge computing: A deep reinforcement learning approach," IEEE Access, vol. 8, pp. 99 098–99 109, 2020.

[13] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," IEEE Transactions on Vehicular Technology, vol. 66, no. 4, pp. 3435–3447, 2016.

[14] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "Noma assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial internet of things," IEEE Transactions on Industrial Informatics, vol. 17, no. 8, pp. 5688–5698, 2020.

[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE/ACM transactions on networking, vol. 24, no. 5, pp. 2795–2808, 2015.

[16] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in 2018 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2018, pp. 1–6.

[17] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," IEEE Transactions on Mobile Computing, vol. 19, no. 11, pp. 2581–2593, 2019.

[18] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, "Efficient offloading for minimizing task computation delay of noma-based multiaccess edge computing," IEEE Transactions on Communications, vol. 70, no. 5, pp. 3186–3203, 2022.

[19] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, "Collaborative task scheduling for iot-assisted edge computing," IEEE Access, vol. 8, pp. 216 593–216 606, 2020.

[20] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things," IEEE Transactions on Industrial Informatics, vol. 17, no. 7, pp. 4925–4934, 2020.

[21] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," ACM Computing Surveys (CSUR), vol. 52, no. 1, pp. 1–23, 2019.

[22] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, "Collaborative cache allocation and computation offloading in mobile edge computing," in 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2017, pp. 366–369.

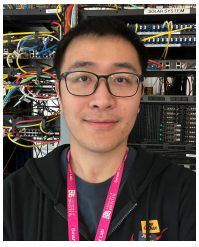
[23] B. S. Baker, "A new proof for the first-fit decreasing bin-packing algorithm," Journal of Algorithms, vol. 6, no. 1, pp. 49–70, 1985.

[24] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Qos-aware mobile edge computing system: Multi-server multi-user scenario," in 2018 IEEE Globecom Workshops (GC Wkshps). IEEE, 2018, pp. 1–6.

[25] M. ETSI, "Mobile edge computing (mec); framework and reference architecture," ETSI, DGS MEC, vol. 3, 2016.

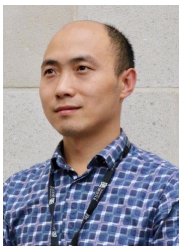
[26] H. Yu, F. Musumeci, J. Zhang, Y. Xiao, M. Tornatore, and Y. Ji, "Du/cu placement for c-ran over optical metro-aggregation networks," in

International IFIP Conference on Optical Network Design and Modeling. Springer, 2019, pp. 82–93.
[27] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," IEEE Transactions on Vehicular Technology, vol. 68, no. 1, pp. 856–868, 2018.
[28] X. Chen, "Decentralized computation offloading game for mobile cloud computing," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 4, pp. 974–983, 2014.
[29] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," IEEE network, vol. 32, no. 1, pp. 54–60, 2018.
[30] M. Schwartz, Mobile wireless communications. IET, 2005, vol. 25.
[31] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb, "Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead," Management science, vol. 54, no. 7, pp. 1336–1349, 2008.
[32] C. C. Bennett and K. Hauser, "Artificial intelligence framework for simulating clinical decision-making: A markov decision process approach," Artificial intelligence in medicine, vol. 57, no. 1, pp. 9–19, 2013.
[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.
[34] D. C. Le and N. Zincir-Heywood, "A frontier: dependable, reliable and secure machine learning for network/system management," Journal of Network and Systems Management, vol. 28, no. 4, pp. 827–849, 2020.
[35] S. S. Gill, S. Tuli, A. N. Toosi, F. Cuadrado, P. Garraghan, R. Bahsoon, H. Lutfiyya, R. Sakellariou, O. Rana, S. Dustdar et al., "Thermosim: Deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments," Journal of Systems and Software, vol. 166, p. 110596, 2020.
[36] D. C. Verma, H. Zhang, and D. Ferrari, Delay jitter control for real-time communication in a packet switching network. International Computer Science Institute, 1991.
[37] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in International conference on machine learning. PMLR, 2014, pp. 387–395.
[38] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," Advances in neural information processing systems, vol. 30, 2017.
[39] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in International Conference on Machine Learning. PMLR, 2017, pp. 2681–2690.
[40] G. Hardin, "Extensions of" the tragedy of the commons"," Science, vol. 280, no. 5364, pp. 682–683, 1998.
[41] H. T. Jongen, T. Möbert, and K. Tammer, "On iterated minimization in nonconvex optimization," Mathematics of operations research, vol. 11, no. 4, pp. 679–691, 1986.
[42] C. Niculescu and L.-E. Persson, Convex functions and their applications. Springer, 2006, vol. 23.
[43] J. Leem and H. Y. Kim, "Action-specialized expert ensemble trading system with extended discrete action space using deep reinforcement learning," Plos one, vol. 15, no. 7, p. e0236178, 2020.



**Haiyuan Li** received the B.Sc degree in communication engineering in Central South University, China, in 2019, and the M.Sc degree in communication networks and signal processing from University of Bristol, U.K., in 2020. He is currently pursuing his Ph.D. degree with the school of electrical and electronic engineering in University of Bristol. His research interests include Mobile Edge Computing, Machine Learning, Game Theory and Network Optimization.

**Karcius Day R. Assis** received the B.Sc. degree in electrical engineering from the Federal University of Paraíba (currently Federal University of Campina Grande) in 1998, the M.Sc. degree from the Federal University of Espírito Santo in 2000, and the Ph.D. degree in electrical engineering from the University of Campinas in 2004. From 2004 to 2010, he was with Salvador University, Federal University of ABC-UFABC and Federal University of Recôncavo da Bahia, Brazil. In 2010, he joined the Department of Electrical and Computer Engineering, Federal University of Bahia, Salvador, Brazil, where he is an Associate Professor. Currently, Karcius has returned to the High Performance Networks Group, Smart Internet Lab at the University of Bristol, U.K., as a visiting researcher, having held this position in 2015. His research interests include Optical Networking, Traffic Engineering, Resource Management, Network Survivability, Machine Learning, Future Internet, and Telecommunication Networks Optimization and Projects. He won an International Visiting Fellowships in 2017 and spent some time at Essex University, United Kingdom.



**Dr Shuangyi Yan** is a Senior Lecturer in the High Performance Networks Group in the Smart Internet Lab at the University of Bristol. He received the B.E degree in information engineering from Tianjin University, Tianjin, China in 2004. In 2009, he got the PhD degree in Optical Engineering from Xi'an Institute of Optics and Precision Mechanics, CAS, Xi'an, China. From 2011 to 2013, Dr Yan worked in the Hong Kong Polytechnic University, Hong Kong, as a postdoctoral researcher, investigating on the spectra-efficient long-haul optical transmission system and low-cost short-range transmission system. In July 2013, he joined the University of Bristol. His research focuses on machine-learning applications in dynamic optical networks and 5G Beyond networks, programmable optical networks, and data centre networks. He is the author or co-author of over 80 refereed publications, of which include several post-deadline papers in optical communication related top-level conferences.

**Dimitra Simeonidou** (FREng, FIEEE) is a Full Professor at the University of Bristol, the Co-Director of the Bristol Digital Futures Institute, and the Director of the Smart Internet Lab. Her research focuses in the fields of high performance networks, programmable networks, wireless-optical convergence, 5G/B5G and smart city infrastructures. She is increasingly working with Social Sciences on topics of digital transformation for society and businesses. Dimitra has also been the Technical Architect and the CTO of the Smart City project Bristol Is Open. She is currently leading the Bristol City/Region 5G urban pilots. She is the author and co-author of over 500 publications, numerous patents and several major contributions to standards. Dimitra is a Fellow of the Royal Academy of Engineering, a Fellow of the Institute of Electrical and Electronic Engineers, and a Royal Society Wolfson Scholar.