Computation Offloading and Resource Allocation in IoT-Based Mobile Edge Computing Systems

Bintao Hu*, Yuan Gao§, Wenzhang Zhang*, Dongyao Jia‡, Hengyan Liu†
*School of Internet of Things, ‡School of Advanced Technology, †School of AI and Advanced Computing
Xi'an Jiaotong-Liverpool University, Suzhou, China
§School of Communication and Information Engineering
Shanghai University, Shanghai, China
*Bintao.Hu, *Wenzhang.Zhang, †Hengyan.Liu, ‡Dongyao.Jia@xjtlu.edu.cn, §gaoyuansie@shu.edu.cn

***Abstract*—With the rise in popularity of artificial intelligence (AI) and internet of things (IoT) technologies, advanced AI technologies have been widely applied to support delay/time-sensitive tasks of IoT-based user equipment (UE) in IoT systems, which allows IoT-based UEs to offload their tasks to a remote fog, edge or cloud computing server. To reduce the consumption of delays (which may include transmission delays, queueing delays, and processing delays) while efficiently allocating the computation resource at a remote server, an efficient offloading decision solution needs to be proposed. In this paper, an IoT-based network system consisting of two layers will be proposed, where The bottom layer is the IoT-based UE layer, which includes multiple IoT-based UEs, and the top layer is the mobile edge computing (MEC) layer, which includes an edge node embedded with the base station. We propose a double Q-Learning-based offloading decision and computation resource allocation optimisation algorithm (DQOCA), which aims to jointly optimise the offloading decisions among all IoT-based UEs and optimise computation resource at the MEC server to reduce the maximum delay consumption among all IoT-based UEs. Simulation findings show that, in comparison to benchmarks (i.e., local processing and edge processing schemes), our proposed approach greatly minimises the maximum delay consumption.**

***Index Terms*—Internet of Things, resource allocation, double Q-learning, edge computing**

I. Introduction

Artificial intelligence (AI)-based Internet of Things (IoT) ideas like the smart home, smart industry, intelligent transportation, industry 4.0, etc., have gained popularity in recent years owing to AI [1]. Due to the capacity limitation of each IoT-based user equipment (UE) to process the task by itself, mobile edge computing (MEC) technologies have been suggested that could provide neighbouring IoT devices additional low-cost computing and storage capabilities with a much lower latency consumption [2]–[4].

To support AI-empowered delay-sensitive tasks of IoT-based UEs, IoT-based UEs which employ MEC technologies have been given the capacity to offload their highly computational and delay-intensive tasks to a distant server (a cloud and/or a MEC edge node) for processing [5]–[7]. An *et al*. [8] proposed a MEC system to minimise each device's consumption of energy under a processing latency constraint while satisfying the specific task requirements. Sheng *et al*. [9] designed a computationally-intensive task-allocation mechanism for an edge computing system to maximise the success ratio with different virtual machines while guaranteeing the average task level of satisfaction based on the suggested task-allocation and resource allocation with the REINFORCE algorithm. Kong *et al*. [10] proposed a k-means clustering optimisation algorithm based on trust weight under a three-layer IoT edge computing system (including a network layer, a broker layer and a device layer) to optimise task offloading and improve the ability of trust computing. Hwang *et al*. [11] designed a virtualization-enabled IoT platform architectural framework to support multiple specific IoT services while guaranteeing low latency and efficient resource and service management. To achieve an optimal application offloading strategy while reducing the cost of the dynamic MEC system, Chen *et al*. [12] proposed an online dynamic task assignment algorithm to accomplish the issue. Hu *et al*. [13] introduced a cloud and fog mixed architecture for vehicular device delay-sensitive tasks offloading scenario, and The corresponding optimisation algorithm was put forth to optimise task offloading decisions and computation resource allocation at a MEC or a cloud server while minimising the maximum local/remote processing latency across all vehicular devices. However, the above studies on cloud computing, fog computing, and MEC computing offloading scenarios have not sufficiently taken into account the overestimated action value and/or convergence speed issues of optimisation algorithms, which may increase the energy consumption and lengthen the delay based on different computation offloading decisions.

In this paper, in order to process IoT-based UE delay-sensitive tasks (i.e., online video calls, online gaming, etc [14], [15]) and guarantee fairness among them successfully, an IoT wireless network supported by an IoT-based two-layer MEC edge computing system. Moreover, we aim to minimise the maximum total delay consumption (which may include the time required for processing tasks, and the delay required for transmitting tasks from an IoT-based UE to a BS) among all IoT-based UEs. To be more specific, we propose a two-layer offloading and computation framework (i.e., IoT-based UEs layer and MEC edge node layer) to evaluate the delay consumption of IoT-based UE tasks pro-

This work is supported in part by XJTLU Research Development Fund under Grant No. RDF-22-02-047, No. RDF-22-02-118, and No. RDF-22-02-099, and in part by supported by the Innovation Program of Shanghai MunicipaScience and Technology Commission under Grant 22511103202.

cessing locally or processing remotely (i.e., offloading the corresponding delay-sensitive tasks to the MEC edge node for further processing), respectively. Based on the defined delay consumption model, the maximum total latency consumption among all IoT-based UEs is to be minimised via a double Q-Learning-based offloading decision and computation resource allocation optimisation algorithm (DQOCA), which optimises the computation resource allocation at the MEC edge node for the remote processing IoT-based UEs based on the optimised delay-sensitive offloading decisions among all IoT-based UEs. By contrasting them with benchmarks such as local processing and edge processing, the performance of the proposed algorithm, DQOCA, according to simulation results is assessed.

II. System Model

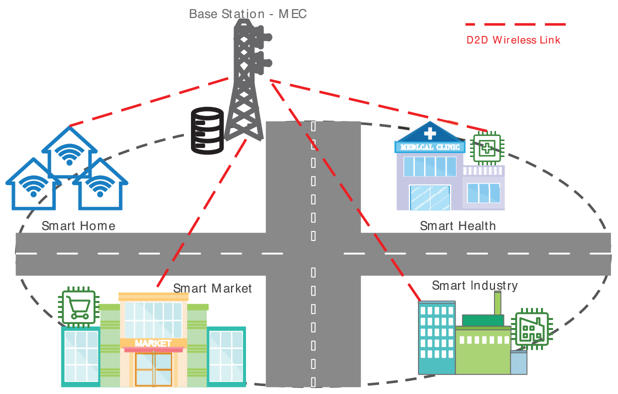

Fig. 1. An IoT-based two-layer MEC edge computing system.

In this section, we first introduce a two-layer IoT-based MEC network system, and then the latency consumption models of the local processing, and edge processing will be derived, respectively.

A. IoT-based MEC Network

We propose a two-layer IoT-based MEC system model as shown in Fig.1, which includes an IoT-based UE layer and a MEC edge node layer. The IoT-based UE layer includes N IoT-based UEs and denotes the set of IoT-based UEs as $\mathcal{N} = \{1, 2, ..., N\}$. In the MEC edge node layer which is composed of a MEC node collected with a base station (BS). All IoT-based UEs are connected to the BS within the communication coverage via device-to-device (D2D) wireless links.

Without losing generality, we assume all tasks are processed by the two-layer IoT-based MEC system with the same length time slots, i,e., Δt. We assume that each IoT-based UE generates only one task at each time slot to be processed and the road-trip time can be neglected [13]. Based on the consideration of delay-sensitive tasks, each IoT-based UE may process its task locally (which is defined as local processing) or the IoT-based UE offloads the task to the MEC edge node for remote processing (which is defined as edge processing). Therefore the corresponding offloading decision for the nth IoT-based UE can be denoted by $l_n, e_n \in \{0, 1\}$, where $l_n = 1$ and $e_n = 1$ illustrate that the delay-sensitive task will be processed by the nth IoT-based UE itself, or by the MEC server, respectively; otherwise, $l_n = 0, e_n = 0$; then, we have

$$l_n + e_n = 1, \forall n \in \mathcal{N}. \tag{1}$$

At each time slot, the offloading decision matrix $\mathbf{I}$ illustrates the offloading decision of all IoT-based UE, which is following by

$$\mathbf{I} = \begin{bmatrix} l_1, & e_1 \\ \vdots & \vdots \\ l_N, & e_N \end{bmatrix}_{N\times 2}, \tag{2}$$

where the nth row indicates the offloading decision for the nth IoT-based UE.

B. Local-Processing Mode

In this mode, if the nth task of IoT-based UE is processed by itself, there is no transmission delay between the UE n and the MEC edge node for this task. Therefore, the service delay only includes processing delay, which can be represented as follows

$$T_n^{loc} = \frac{D_n C_n}{f_n^{loc}}, \tag{3}$$

where C_n (in CPU cycles/bit) denotes the processing density of the nth task; the nth IoT-based UE's local processing capacity is denoted by f_n^{loc} (in CPU cycles/s); and the input data size of the nth task is denoted as D_n (in bits).

C. Edge-Processing Mode

In this mode, if the task is to be processed by a MEC edge node (i.e., $e_n = 1$), the nth task will be uploaded to the MEC edge node first, and then the task is to be processed based on the allocated computation resource at the MEC edge node. Therefore, the service delay for edge-processing is composed of the transmission delay and processing delay.

1) Transmission Delay: If the task needs to be uploaded to the MEC server, for simplicity, we assume that each task is offloaded over a selected resource block (RB), which allows us to ignore the communication resource allocation issue [13]. Thus, the maximum achievable uplink transmission rate from the nth IoT-based UE to the BS over a selected RB can be expressed as

$$r_{n,e} = W log_2 \left(1 + \frac{p_n g_{n,e}}{N_0}\right), \tag{4}$$

where W (in Hz) denotes the bandwidth of the allocated RB; p_n (in mW) denotes the transmission power of IoT-based UE n; $g_{n,e}$ indicates the channel power gain between the nth IoT-based UE and the BS; N_0 (in dBm/Hz) denotes the additive white Gaussian noise (AWGN) power at the BS.

The transmission delay from the nth IoT-based UE to the BS is given by

$$T_{n,e}^{tran} = \frac{D_n}{r_{n,e}}. \tag{5}$$

2) Processing Delay: Since all the data of the nth task has been uploaded to the BS, the task starts to be processed at the MEC edge node, and the nth task's processing latency at the MEC edge node can be given by

$$T_{n,e}^{edge} = \frac{D_n C_n}{f_n^{edge}}, \tag{6}$$

where f_n^{edge} (in CPU cycles/s) is the allocated computation resource at the MEC edge node to the nth IoT-based UE.

3) Service Delay: According to (5) and (6), the delay consumption for processing task n at the MEC edge node is given by

$$T_n^{edge} = T_{n,e}^{tran} + T_{n,e}^{edge}. \tag{7}$$

III. Problem Formulation and Proposed Algorithm

In this section, the proposed IoT-based MEC system is applied to produce the min-max latency consumption issue, which is subsequently solved using a double Q-learning-based offloading decision and computation resource allocation optimisation algorithm (DQOCA).

A. Problem Formulation

The delay consumption for each task may encompass the processing delay and/or the transmission delay, depending on its respective offloading decision. Thus, the delay consumption of the nth task can be given by

$$T_n = l_n T_n^{loc} + e_n T_n^{edge}, \tag{8}$$

where T_n^{loc} and T_n^{edge} are given in (3) and (7), respectively.

We optimise computation resource allocation $\mathbf{f}^{edge}$ at the MEC edge node and the offloading decisions $\mathbf{I}$ in an effort to reduce the maximum total latency consumption of all IoT-based UEs. It is possible to phrase the offloading decision of all IoT-based UEs and the computation resource allocation at the MEC server optimisation problem as follows

$$\mathcal{P}: \min_{\mathbf{I},\mathbf{f}^{edge}} \max_{n\in\mathcal{N}} T_n \tag{9}$$

$$\begin{aligned}
s.t.\quad & l_n, e_n \in \{0,1\},\ \forall n \in \mathcal{N}, && \text{(9a)}\\
& l_n + e_n = 1,\ \forall n \in \mathcal{N}, && \text{(9b)}\\
& \sum_{n\in\mathcal{N}_1} e_n f_n^{edge} \le F^{edge}, && \text{(9c)}\\
& 0 \le f_n^{loc} \le f_n^{edge},\ \forall n \in \mathcal{N}, && \text{(9d)}\\
& p_n \le p_{max},\ \forall n \in \mathcal{N} && \text{(9e)}
\end{aligned}$$

where $\mathcal{N}_1$ is the set of edge-processing IoT-based UEs; F^{edge} denotes the maximum computation capability in the MEC edge server; (9a) denotes the offloading decision indicator for each IoT-based UE; (9b) indicates that each task is able to be processed either locally by the IoT-based UE or by the MEC server; (9c) indicates that the total computation resources allocated at the MEC edge node must not surpass its maximum computation capability; (9d) is the constraint on allocated computation capacity, where the amount of computation resource available at the MEC edge node is larger than the available for local processing, while both of them should be non-negative; and (9e) is the constraint on the transmit power of each IoT-based UE.

B. Deep Learning-Based Markov Decision Process Model

In order to resolve the formulated problem (9) according to the deep learning optimisation algorithm, which can be reformulated as a Markov decision process (MDP) model [16], and the corresponding MDP model can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, which includes four components (i.e., state space, action space, state-transition probability, and reward function, respectively).

1) State Space: The state space is denoted as $\mathcal{S} = \{s(t), t \in \mathcal{T}\}$, where $s(t)$ indicates state at the tth time slot, which is given by

$$s(t) \triangleq \{\mathbf{D}(t), \mathbf{C}(t), \mathbf{r}(t), \mathbf{f}^{edge}(t)\}, \tag{10}$$

where $\mathbf{D}(t)$ is the set of the input data of each IoT-based UE at the tth time slot; $\mathbf{C}(t)$ is the set of the required total number of CPU cycles of each task at the tth time slot; $\mathbf{r}(t)$ denotes the set of the achievable uplink transmission rate between each IoT-based UE and the BS at the tth time slot; and $\mathbf{f}^{edge}(t)$ is the available computation capacity at the MEC edge node.

2) Action Space: The set $\mathcal{A} = \{a(t), t \in \mathcal{T}\}$ denotes the action space, where $a(t)$ indicates action, which includes offloading decision action and computation resource allocation action at the tth time slot. Therefore, at the tth time slot the action can be expressed by

$$a(t) \triangleq \{\mathbf{l}(t), \mathbf{e}(t), \mathbf{f}(t)\}, \tag{11}$$

where $\mathbf{l}(t)$ indicates the set of the local-processing decision if the nth task is to be processed by itself; $\mathbf{e}(t)$ indicates the set of the edge-processing decision if the nth task is to be processed at the MEC edge node, and $\mathbf{f}(t)$ is the set of the allocated computation resource to each IoT-based UE at the tth time slot.

3) Policy Space: A policy is a mapping operator from state space to action space, which denotes $\pi(a(t)|s(t)) : \mathcal{S} \to \mathcal{A}$.

C. Double Q-Learning-Based Offloading Decision and Computation Resource Allocation Optimisation Algorithm

Traditional Q-learning algorithms aim to minimise rewards to achieve target tasks based on the proposed environment [17], the action-state value function can be given by

$$Q_\pi(s(t), a(t)) = \mathbb{E}\left[R(t+1) + \beta R(t+2) + ...|s_0 \in \mathcal{S}, a_0 \in \mathcal{A}\right], \tag{12}$$

where $\beta \in [0,1]$ is the discount factor. In order to update the Q-table, which is given by

$$Q(t+1)(s(t),a(t)) = Q(t)(s(t),a(t)) + \alpha\big[R(t)+ \beta \max_a Q(t)(s(t+1),a) - Q(t)(s(t),a(t))\big], \tag{13}$$

where $\alpha \in (0,1]$ denotes the learning rate, and the action with maximum $\max_a Q(t)(s(t+1),a)$ will be selected.

For most of the existing methods may result in Q-learning overestimated action value issues [18]. To overcome this challenge, we propose a DQOCA to deal with the problem (9), as summarised in Algorithm 1. We first initialise two learning networks Q^X and Q^Y based on the environment. Comparable to the classic Q-learning algorithm at each learning iteration, the Q-table for Q^X is to be updated according to $Q^Y(t+1)(s(t),a^*)$, which can be given by

$$\begin{aligned} Q^X(t+1)(s(t),a(t)) &= Q^X(t)(s(t),a(t)) + \alpha(s(t),a(t)) \\ &\big[R(t) + \beta Q^Y(t)(s(t+1),a^*) \\ &- Q^X(t)(s(t),a(t))\big], \end{aligned} \tag{14}$$

where $a^* = arg\max_a Q^X(t{+}1)(s(t),a)$. Similarly, the Q-table for Q^Y can be updated as follow

$$\begin{aligned} Q^Y(t+1)(s(t),a(t)) &= Q^Y(t)(s(t),a(t)) + \alpha(s(t),a(t)) \\ &\big[R(t) + \beta Q^X(t)(s(t+1),a^*) \\ &- Q^Y(t)(s(t),a(t))\big]. \end{aligned} \tag{15}$$

For each action, Q-values for the Q^X network and Q^Y network can be averaged accordingly, and the further selection process is to follow a greedy policy [19]. When the training achieves the target or the number of iterations allowed that are permitted, the optimal MEC edge node computation resource allocation $\mathbf{f}^{edge*}$ and the optimal offloading decision $\mathbf{I}^*$ will be returned.

Algorithm 1 Double Q-learning-Based Offloading Decision and Computation Resource Allocation Optimisation Algorithm (DQOCA)

```
1: Initialise $t = 0$, the discount factor $\beta$, the learning rate $\alpha$,
   and $Q^X(a) = Q^X(a) = 0, \forall a \in \mathcal{A}$.
2: Reset the environment with an arbitrary observation $s$.
3: repeat
4:   Select action according to $Q^X(s,\cdot), Q^Y(s,\cdot)$, and $s'$.
5:   Update either $Q^X$ or $Q^Y$ based on a $\epsilon-$greedy policy
     [19].
6:   if update $Q^X$ then
7:     Define $a^* = arg\max_a Q^X(s',a)$.
8:     $Q^X(s,a)$ is evaluated according to (14).
9:   else
10:    Define $a^* = arg\max_a Q^Y(s',a)$.
11:    $Q^Y(s,a)$ is evaluated according to (15).
12:  end if
13:  $s \leftarrow s'$.
14: until end
15: return $\mathbf{I}^*, \mathbf{f}^{edge*}$.
```

TABLE I
SIMULATION PARAMETERS [13], [16], [20]

Parameters	**Value**
Number of IoT-based UEs V	5
Number of MEC edge node	1
Transmit bandwidth, W	10 MHz
Transmit power of IoT-based UE n, p_n	200 mW
The AWGN power density at the MEC edge node, N_0	-174 dBm/Hz
Data size of a task of IoT-based UE n, D_n	$[300, 500]$ kbits
Local-processing capability of IoT-based UE n, f_n^{loc}	1 G cycles/s
Total computation capability of the MEC edge node, $\mathbf{f}^{edge}$	5 G cycles/s
Learning rate, α	0.01
Discount factor, β	0.8

IV. Simulation Results

In this section, the simulation results with the two-layer MEC edge computing system as shown in Fig.1 that has been provided and analysed. We assume a single-cell network with one MEC edge node embedded BS at the edge of a $1000m \times 1000m$ region. Additionally, we make the assumption that each IoT-based UE is uniformly distributed throughout the proposed region. For each IoT-based UE, there is only one task to be processed at the tth time slot, which cannot be divided into multiple sub-task. Unless specified otherwise, the values for each parameter used in the simulation work are listed in the TABLE I [13], [16], [20].

As shown in Fig. 2, the maximum total delay consumption versus the number of IoT-based UEs, where the proposed algorithm1 ('DQOCA'), 'Local-Processing', and 'Edge-Processing' indicate that the scenarios where all delay-sensitive tasks of the IoT-based UEs will be processed locally, or by the MEC edge node, respectively. As Fig. 2 indicates that the maximum total latency consumption increases with the number of IoT-based UEs among all the involved scenarios. In addition, our proposed optimisation algorithm, DQOCA, performs the best compared among all the considered scenarios for any given number of IoT-based UEs. If the number of IoT-based UEs is larger than 5, the edge-processing case performs the highest maximum total service delay among all proposed scenarios. It is because all the edge-computing tasks share the limited computation capacity of the MEC edge node.

The connection between the maximum total delay consumption and the maximum input data size D_n for each task is shown in Fig. 3, where $N = 3$. Additionally, Fig. 3 also shows that the overall maximum delay consumption in each scenario analysed is higher the larger the data size of each task. The DQOCA algorithm we've proposed performs the best out of all the cases we've taken into account. The enormous volume of input data for each task also has the greatest influence on local processing because of the restricted computing capacity

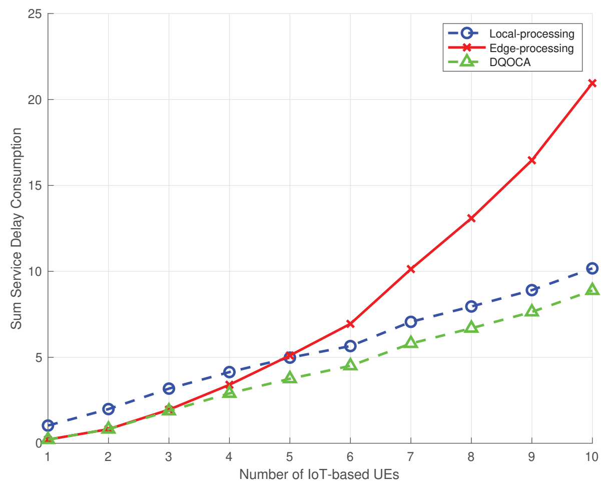

Fig. 2. Maximum total delay consumption versus the number of IoT-based UEs.

of each IoT-based UE.

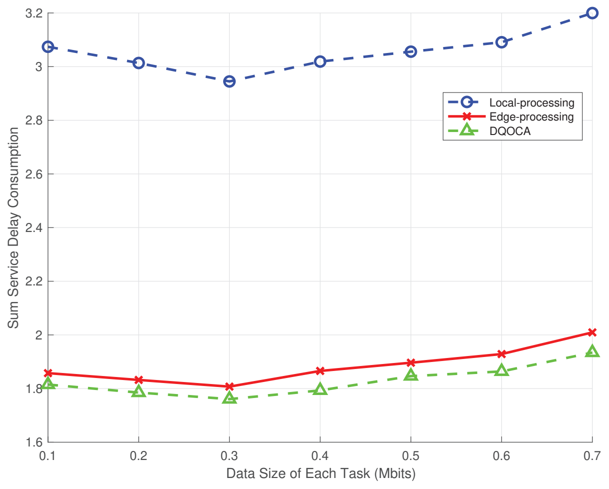

Fig. 3. Maximum total delay consumption vs. the data size of the task.

V. Conclusion

In this paper, we proposed a double Q-learning-based offloading decision and computation resource allocation optimisation algorithm, which optimised the offloading decisions among all IoT-based UEs while optimising the computation resource allocation optimisation at the MEC edge node. The proposed algorithm minimised the maximum total delay consumption of all IoT-based UEs according to a two-layer MEC edge computing system. Based on the proposed system, each IoT-based UE can offload its corresponding latency-sensitive tasks to the MEC edge node or process it locally. The simulation results demonstrate that the suggested approach, DQOCA, consumes a maximum total latency that is significantly less than that of local processing and edge processing.

References

[1] Z. Ali, Z. H. Abbas, G. Abbas, A. Numani, and M. Bilal, "Smart computational offloading for mobile edge computing in next-generation internet of things networks," *Computer Networks*, vol. 198, p. 108356, 2021.

[2] S. Naveen and M. R. Kounte, "Key technologies and challenges in iot edge computing," in *2019 Third international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)*. IEEE, 2019, pp. 61–65.

[3] K.-H. Le, K.-H. Le-Minh, and H.-T. Thai, "Brainyedge: An ai-enabled framework for iot edge computing," *ICT Express*, 2021.

[4] Y. Yang, X. Luo, X. Chu, and M.-T. Zhou, *Fog-enabled intelligent IoT systems*. Springer, 2020.

[5] 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.

[6] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *Journal of Systems Architecture*, vol. 118, p. 102225, 2021.

[7] C. Jiang, Y. Li, J. Su, and Q. Chen, "Research on new edge computing network architecture and task offloading strategy for internet of things," *Wireless Networks*, pp. 1–13, 2021.

[8] X. An, R. Fan, H. Hu, N. Zhang, S. Atapattu, and T. A. Tsiftsis, "Joint task offloading and resource allocation for iot edge computing with sequential task dependency," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 546–16 561, 2022.

[9] S. Sheng, P. Chen, Z. Chen, L. Wu, and Y. Yao, "Deep reinforcement learning-based task scheduling in iot edge computing," *Sensors*, vol. 21, no. 5, p. 1666, 2021.

[10] W. Kong, X. Li, L. Hou, J. Yuan, Y. Gao, and S. Yu, "A reliable and efficient task offloading strategy based on multifeedback trust mechanism for iot edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 927–13 941, 2022.

[11] J. Hwang, L. Nkenyereye, N. Sung, J. Kim, and J. Song, "Iot service slicing and task offloading for edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 526–11 547, 2021.

[12] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 421–432, 2022.

[13] B. Hu, J. Du, and X. Chu, "Enabling low-latency applications in vehicular networks based on mixed fog/cloud computing systems," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 722–727.

[14] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "Mec-assisted immersive vr video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9517–9529, 2020.

[15] B. Hu and X. Chu, "Social-aware resource allocation for vehicle-to-everything communications underlaying cellular networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, 2021, pp. 1–6.

[16] J. Du, W. Cheng, G. Lu, H. Cao, X. Chu, Z. Zhang, and J. Wang, "Resource pricing and allocation in mec enabled blockchain systems: An a3c deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 33–44, 2021.

[17] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[18] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, 2010.

[19] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," in *KI 2010: Advances in Artificial Intelligence: 33rd Annual German Conference on AI, Karlsruhe, Germany, September 21-24, 2010. Proceedings 33*. Springer, 2010, pp. 203–210.

[20] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2017.