Zhang *et al.*
EURASIP Journal on Advances in Signal Processing (2022) 2022:45
https://doi.org/10.1186/s13634-022-00876-1

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

DQN-based mobile edge computing for smart Internet of vehicle

Lianhong Zhang[1], Wenqi Zhou[1], Junjuan Xia[1*], Chongzhi Gao[1*], Fusheng Zhu[2*], Chengyuan Fan[3] and Jiangtao Ou[3]

*Correspondence: xiajunjuan@gzhu.edu.cn; czgao@gzhu.edu.cn; zhufusheng@gdcni.cn

[1] School of Computer Science, Guangzhou University, Guangzhou, China
[2] Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China
Full list of author information is available at the end of the article

Abstract

In this paper, we investigate a multiuser mobile edge computing (MEC)-aided smart Internet of vehicle (IoV) network, where one edge server can help accomplish the intensive calculating tasks from the vehicular users. For the MEC networks, most existing works mainly focus on minimizing the system latency to guarantee the user's quality of service (QoS) through designing some offloading strategies, which, however, fail to consider the pricing from the server and hence fail to take into account the budget constraint from the users. To address this issue, we jointly incorporate the budget constraint into the system design of the MEC-based IoV networks and then propose a joint deep reinforcement learning (DRL) approach combined with the convex optimization algorithm. Specifically, a deep Q-network (DQN) is firstly used to make the offloading decision, and then, the Lagrange multiplier method is employed to allocate the calculating capability of the server to multiple users. Simulations are finally presented to demonstrate that the proposed schemes outperform the conventional ones. In particular, the proposed scheme can effectively reduce the system latency by up to 56% compared to the conventional schemes.

Keywords: Internet of vehicle, Mobile edge computing, Budget, Offloading strategy, Latency

1 Introduction

With the development of wireless communication technologies [1–4], an increasing number of vehicles are connected to the Internet through access points, which promotes the emergence of the Internet of vehicle (IoV) and smart city [5–7]. In the IoV systems, mobile vehicles can collect data and exert some intensive calculating tasks to make intelligent decisions [8–10]. However, due to the lack of calculating capability in mobile vehicles, local computing may cause a severe latency and even lead to some serious consequences, e.g., traffic accidents [11, 12]. By utilizing the remote calculating resources at the cloud server, the calculating latency can be reduced effectively, at the cost of an increased communication latency [13, 14]. In particular, the communication latency may dominate the system performance when the wireless channels are vulnerable. Moreover, massive requests arising from the network edge impose a heavy burden on the cloud server, which further deteriorates the quality of experience (QoE) of users.

Springer Open

© The Author(s) 2022. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

To address the above issues of cloud computing, a novel communication and computation paradigm named mobile edge computing (MEC) was proposed. By deploying calculating access points (CAPs) at the network edge, the calculating tasks can be unpacked to the neighboring CAPs through reasonable task partition and offloading in order to achieve a low latency and energy consumption [15, 16]. To achieve the same goal, the authors in [17] studied a multiuser MEC network, where a deep Q-network (DQN)-based offloading strategy was proposed for the task offloading. With the same reinforcement learning-based method, the authors in [18] focused on the study of channels to obtain better performances. Moreover, the system cost was studied in [19, 20] in terms of a combination of energy consumption and latency, where a joint optimization method of offloading decision and resource allocation was proposed to enhance the network performance. In further, a deep deterministic policy gradient (DDPG) was proposed to resolve the offloading strategy design in the MEC system [21], where a long-term optimization was used. The authors in [22] considered parked vehicles as the computing service providers and proposed a dynamic pricing strategy in order to maximize the revenue of the computing service providers and meanwhile minimize the energy consumption of smart user equipment (UEs). It has been shown in [23–25] that the allocation of channel resources could be optimized to improve the performance of networks, and the authors in [26] enhanced the system performance through a dynamic game model.

The above literature review shows that most of the existing works attempted to optimize the system performance of MEC networks through offloading strategy design and resource allocation. To the best of our knowledge, seldom works have considered the pricing from the server and taken into account the budget constraint from the users. In practice, the pricing from the server may affect the system performance of the MEC networks, as this can affect the calculating capability allocated to the users by the CAPs. In addition, the budget of the users may also affect the network performance, as some users may not have enough budget to buy the computing resources at the CAPs, and the intensive calculating tasks have to be computed locally. Due to these reasons, we will jointly incorporate the budget constraint into the system design of the MEC-based IoV networks in this paper.

In this paper, we investigate a multiuser MEC-aided smart IoV network, where one edge server can help accomplish the intensive calculating tasks from the vehicular users. For the MEC networks, most existing works mainly focus on minimizing the system latency to guarantee the user's quality of service (QoS) through designing some offloading strategies, which, however, fail to consider the pricing from the server and hence fail to take into account the budget constraint from the users. To address this issue, we jointly incorporate the budget constraint into the system design of the MEC-based IoV networks and then propose a joint deep reinforcement learning (DRL) approach combined with the convex optimization algorithm. Specifically, a deep Q-network (DQN) is firstly used to make the offloading decision, and then, the Lagrange multiplier method is employed to allocate the calculating capability of the server to multiple users. Simulations are finally presented to demonstrate that the proposed schemes outperform the conventional ones. In particular, the proposed scheme can effectively reduce the system latency up by to 56% compared to the conventional schemes. The main contributions of this paper are as follows:

- We study a MEC network for IoV, where we not only consider the resources allocation but also combine the charging rules with the users' budget constraints to optimize the performance of the MEC.
- We propose a DQN and convex optimization algorithm, which a convex optimization method is integrated into the DQN framework. This algorithm not only has the advantages of reinforcement learning, but also uses the convex optimization method to reduce the complexity of algorithm and help convergence.
- Simulations show that the proposed DQN and convex optimization algorithm can outperform conventional methods and can effectively reduce the system latency by up to 56%.

The rest of this paper is organized as follows. After Introduction, we discuss the system model of MEC-based IoV network and then present the optimization problem formulation in Sec. 2. After that, we give the DQN and convex optimization algorithm-based method to solve the optimization problem in Sec. 3. We further provide some simulations and discussions in Sec. 4 and, finally, make some conclusions in Sec. 5.

2 Methods/experimental

Figure 1 shows a vehicular MEC network with one CAP and M vehicular users denoted by $\{u_m|1 \leq m \leq M\}$. The users have some latency-sensitive calculating tasks. The CAP can help compute some parts of the tasks with its much more powerful capability, while the other can be computed locally. Moreover, when the tasks are offloaded to the CAP for computing, the CAP will charge users according to the amount of offloading tasks and the calculating capability. The following subsections will introduce the local computing model, the offloading model, and the purchase model, respectively. After that, we will give the system optimization problem.

2.1 Local computing model

As mentioned before, some parts of the tasks can be computed locally, and the local calculating latency is written as

$$T_m^{\text{local}} = \frac{l_m(1-\alpha_m)C}{F_m}, \tag{1}$$

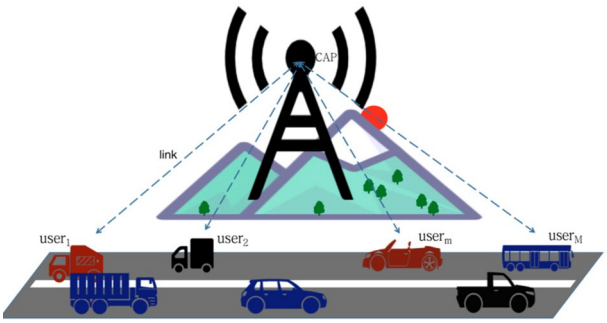

Fig. 1 System model of the considered vehicular MEC

where $\alpha_m \in [0,1]$ is the offloading ratio denoting the ratio of task offloaded from user u_m to the CAP, l_m is the task size of user u_m, C represents the number of required CPU cycles for processing one bit of task, and F_m represents the calculating capability of user u_m measured by its CPU cycle frequency.

2.2 Offloading model

When the tasks are partially offloaded to the CAP for computing, the users should transmit the offloaded tasks through wireless links, and the data transmission rate is [27–29]

$$R_m = W \log_2 \left(1 + \frac{P_m |h_m|^2}{\sigma^2}\right), \tag{2}$$

where W is the bandwidth of the wireless link between user u_m and CAP, P_m represents u_m' transmit power, $h_m \sim \mathcal{CN}(0,\beta)$ represents the channel parameter of the wireless channel, and σ^2 represents the variance of additive white Gaussian noise (AWGN) at the CAP. And then, the communication latency can be written as

$$T_m^{\text{tran}} = \frac{l_m \alpha_m}{R_m}. \tag{3}$$

After receiving the tasks from user u_m, the CAP begins to compute the tasks, and the calculating latency at the CAP is

$$T_m^{\text{CAP}} = \frac{l_m \alpha_m C}{f_m}. \tag{4}$$

in which f_m is the calculating capability that u_m buys from the CAP, and it should satisfy the constraint $\sum_{m=1}^{M} f_m \le F$, in which F is the total calculating capability at the server.

Then, we can get the offloading latency, including the communication latency and the calculating latency at the CAP,

$$T_m^{\text{offl}} = T_m^{\text{tran}} + T_m^{\text{CAP}}. \tag{5}$$

Since the local computation and the offloading are two operations that are executed concurrently, the total calculating latency of u_m is

$$T_m^{\text{total}} = \max\{T_m^{\text{local}}, T_m^{\text{offl}}\}. \tag{6}$$

For the whole vehicular MEC network, users do the local computation and the offloading in parallel. Therefore, the system latency can be defined as the finish time of the tasks from all users,

$$T^{\text{total}} = \max\{T_1^{\text{total}}, T_2^{\text{total}}, T_3^{\text{total}}, \ldots, T_M^{\text{total}}\}. \tag{7}$$

2.3 Purchase model

Note that user u_m needs to pay the CAP when it offloads the tasks to the CAP for computation, and the charging rule of CAP is composed of basic service fees and calculating fees. Specifically, the basic service fee is based on the size of offloading tasks,

and the calculating fee is based on the calculating capability that the CAP allocates to the users. Hence, the payment of user u_m for offloading is

$$U_m = \eta_l l_m \alpha_m + \eta_f f_m, \tag{8}$$

where η_l is the price per bit of the task and η_f is the price paid for the CAP capability. As the budget of each user is limited in practice, so we can get the budget constraint of user u_m as

$$U_m \leq U_m^{\max}, \tag{9}$$

where $U_m^{\max}$ is the maximum budget of user u_m.

2.4 Problem formulation

In practice, the vehicular MEC network involves latency-critical tasks for the dynamic changes in the vehicles [30], and the system needs to process the latency-sensitive tasks from the users as quickly as possible due to the movement of the vehicular users. Therefore, the optimization problem of the network is to minimize the system latency, which can be formulated as

$$\begin{aligned} \mathbf{P1}: \min_{\{\alpha_m, f_m\}} \quad & T^{\text{total}} \\ s.t. \quad C_1: \quad & \alpha_m \in [0,1], \forall m \in [1,M], \\ C_2: \quad & \sum_{m=1}^{M} f_m \leq F, \\ C_3: \quad & U_m \leq U_m^{\max}, \forall m \in [1,M], \end{aligned} \tag{10}$$

where C_1 is the constraint of the offloading ratio, which indicates how many parts the user u_m offloads to the CAP. Constraint C_2 presents that the calculating capability at the server distributed to user u_m may not surpass the total calculating capability. Constraint C_3 denotes that users' payment of offloading should meet the budget constraint. From (10), the offloading ratio and calculating capability can be optimized to minimize the network latency while meeting the budget requirement. However, the optimization problem is complicated and hard to be solved by conventional convex optimization methods. Therefore, we turn to propose a DQN and convex optimization algorithm to resolve the problem. All notations used in this section are summarized in Table 1.

3 DQN and convex optimization algorithm

This section introduces a DQN and convex optimization algorithm for **P1** in (10). The proposed algorithm overcomes the complicated action space caused by the full utilization of DQN which leads to an extremely high cost to perform exploration and then affects the final training result. In the following subsections, we first describe how to obtain the offloading decision through the DQN and then give the process of the resource allocation through the convex optimization method.

Table 1 Symbol notations

Notation	Definition
u_m	The mth user
T_m^{local}	Local latency of the mth user
l_m	Task size of the mth user
α_m	Offloading ratio of the mth user
C	CPU cycle of processing one bit data
F_m	calculating capability of the mth user
R_m	Data transmission rate from u_m to the CAP
W	Wireless bandwidth
P_m	Transmit power of the mth user
h_m	Channel parameter from u_m to the CAP
σ^2	Variance of additive white Gaussian noise (AWGN) at the CAP
T_m^{tran}	Transmission latency of the mth user
T_m^{CAP}	calculating latency at CAP of the mth user
f_m	calculating capability at CAP of the mth user
F	Total calculating capability at the server
T_m^{offl}	Offloading latency of the mth user
T_m^{total}	Total latency of the mth user
T^{total}	Total latency of the vehicular MEC system

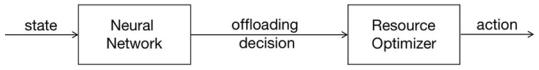

Fig. 2 DQN and convex optimization algorithm

3.1 DQN-based offloading decision

To solve the problem **P1**, we propose the DQN and convex optimization algorithm to obtain the offloading decision and calculating capability allocation. As shown in Fig. 2, the proposed algorithm is composed of a DQN-based method and a convex optimization method. Specifically, we first employ the DQN-based method to obtain the offloading decision. After the offloading decision is obtained, the convex optimization method is used to obtain the allocation decision of calculating capability.

We can model the problem of offloading decisions as Markov decision process (MDP). In MDP, the agent firstly gets the state $s_t \in \boldsymbol{S}$ from the environment at time slot t and then makes an action a_t according to policy π. After that, the agent acts on a_t in environment, causing the state of environment transits from s_t to s_{t+1}, and the agent gets r_t as a reward. Specifically, we define state space as $\boldsymbol{S} = \{\boldsymbol{\alpha}\}$, where $\boldsymbol{\alpha} = \{\alpha_1(t), \alpha_2(t), \alpha_3(t), \ldots, \alpha_M(t)\}$ is the offloading ratio of users at the time slot t, and the action space is $\boldsymbol{A} = \{\rho_1, \rho_2, \rho_3, ..., \rho_m, \ldots, \rho_M, \rho_1^*, \rho_2^*, \rho_3^*, ..., \rho_m^*, \ldots, \rho_M^*\}$, where $\rho_m = -\delta$ and $\rho_m^* = +\delta$ are actions to adjust the offloading ratio under the constraint C_1. Moreover, the reward of the offloading decision problem is related to the system latency [31, 32]

$$r_t = \begin{cases} \tau_1 & \text{If} \quad T^{\text{total}}(t) < T^{\text{total}}(t-1), \\ -\tau_2 & \text{If} \quad T^{\text{total}}(t) = T^{\text{total}}(t-1), \\ -\tau_1 & \text{If} \quad T^{\text{total}}(t) > T^{\text{total}}(t-1), \end{cases} \tag{11}$$

where τ_1 and τ_2 are two positive values with $\tau_1 > \tau_2$. In further, we evaluate the policy π through Q function $Q_\pi(s, a)$, which represents the accumulative rewards from an action a acting on state s. According to Q function, the best policy is

$$\pi^* = \arg \max_{\pi} Q_\pi(s, a), \tag{12}$$

and the agent gets the environment state s which helps in choosing an action through the best policy π^*,

$$a = \arg \max_{a \in \boldsymbol{A}} Q_{\pi^*}(s, a). \tag{13}$$

Based on the above processing, we adopt deep Q-network (DQN), and it uses deep neural networks to approximate the optimal Q function. There are two neural networks in DQN, including the actor-network and the target-network. The role of actor-network is to predict the action $a_t \in A$ by inputting the state $s_t \in \boldsymbol{S}$. Generally, $Q(s, a)$ is used to denote $Q_{\pi^*}(s, a)$. To avoid the offloading optimization problem falling into a local optimal value, we can obtain the action by the ϵ-greedy policy,

$$a_t = \begin{cases} \arg \max_{a \in \boldsymbol{A}} Q(s_t, a; \boldsymbol{\theta}), & \text{with probability } \epsilon, \\ \text{Randomly choose from } \boldsymbol{A}, & \text{otherwise}, \end{cases} \tag{14}$$

where $\boldsymbol{\theta}$ is the weights of the actor-network. In order to better approximate the Q function, we adopt the temporal difference (TD) approach in DQN,

$$Q(s_t, a_t; \boldsymbol{\theta}) = r_t + \gamma \max_{a \in \boldsymbol{A}} (Q(s_{t+1}, a; \boldsymbol{\theta})). \tag{15}$$

To obtain the TD target of DQN, we add a target-network as a copy of target-network, which is reset as the actor-network every T_u time slots. And the loss function is [33–35]

$$L_t = ((r_t + \gamma \max_{a \in \boldsymbol{A}} (Q(s_{t+1}, a; \hat{\boldsymbol{\theta}}))) - Q(s_t, a_t; \boldsymbol{\theta}))^2, \tag{16}$$

where $\hat{\boldsymbol{\theta}}$ is the weights of the target-network. And we use a back-propagation (BP) algorithm to update the actor-network every T_l time slots. To break the relationship between data created at every time slot, we adopt an experience replay (ER) and a mini-batch sampling. A transition (s_t, a_t, r_t, s_{t+1}) is stored into ER at each time slot, and a mini-batch size of transitions are randomly sampled from ER to update the actor-network with BP algorithm every T_l time slots.

3.2 Convex optimization-based resource allocation

After obtaining the offloading decision α_m by the DQN-network, the problem **P1** can be transformed into

$$\begin{aligned} \mathbf{P2}: \min_{\{f_m\}} \quad & T^{\text{total}} \\ s.t. \quad C_1: \quad & \sum_{m=1}^{M} f_m \le F, \\ C_2: \quad & U_m \le U_m^{\max}, \forall m \in [1, M]. \end{aligned} \tag{17}$$

From **P2**, we can observe that the minimization problem is affected by the total calculating capability at the server and the budget of the user. The feasibility of the solution of the whole optimization **P1** highly depends on the training and the design of DQN. As a part of the designed DQN, the solution of **P2** has a limited impact on the training of the DQN. Moreover, a powerful DQN can still get a reliable and feasible solution to the whole problem even **P2** cannot obtain the optimal solution. Therefore, it is worthwhile to find a solution with a lower complexity for **P2**. Hence, we firstly limit the capability allocated to users in their budget and then based on the limit, we get the solution constraints by the total capability at the server shared by all users. To this end, we firstly relax the constraint C_2 and transform **P2** into a convex problem,

$$\begin{aligned} \mathbf{P3}: \min_{\{f_m\}} \quad & T^{\text{total}} \\ s.t. \quad C_1: \quad & \sum_{m=1}^{M} f_m \le F. \end{aligned} \tag{18}$$

Then, we adopt the Lagrange multiplier method to optimize the problem **P3**, and the Lagrange function can be written as

$$\mathscr{L}(f_m, \lambda) = \sum_{m=1}^{M} T_m^{\text{CAP}} + \lambda\left(\sum_{m=1}^{M} f_m - F\right), \tag{19}$$

where $\lambda > 0$ is a Lagrange multiplier. From (19), we set the first partial derivative of $\mathscr{L}(f_m, \lambda)$ with respect to f_m and λ to zero,

$$\begin{cases} \frac{\partial \mathscr{L}}{\partial f_m} = -\frac{l_m \alpha_m}{f_m^2} + \lambda = 0, \\ \frac{\partial \mathscr{L}}{\partial \lambda} = \sum_{m=1}^{M} f_m - F = 0. \end{cases} \tag{20}$$

By combining and solving the above two equations, we can obtain the optimal solution of **P3** as

$$\sqrt{\lambda} = \frac{1}{F} \sum_{m=1}^{M} \sqrt{l_m \alpha_m}, \tag{21}$$

$$f_m = \frac{l_m \alpha_m F}{\sum_{m_1=1}^{M} \sqrt{l_{m_1} \alpha_{m_1}}}. \tag{22}$$

After obtaining the optimal solution of the relaxed problem **P3**, we further consider the constraint C_2 and give a feasible solution for **P2**. According to (8) and (9), we can obtain

$$0 \le f_m \le \frac{U_m^{\max} - \eta_l l_m \alpha_m}{\eta_f}. \tag{23}$$

By jointly considering (22) and (23), we can finally obtain a feasible solution of **P2**,

$$f_m = \begin{cases} \dfrac{l_m \alpha_m F}{\sum_{m=1}^{M} \sqrt{l_m \alpha_m}}, & \text{If } \dfrac{l_m \alpha_m F}{\sum_{m=1}^{M} \sqrt{l_m \alpha_m}} \le \dfrac{U_m^{\max} - \eta_l l_m \alpha_m}{\eta_f}, \\ \dfrac{U_m^{\max} - \eta_l l_m \alpha_m}{\eta_f}, & \text{Else.} \end{cases} \tag{24}$$

From the above description, we can summarize the procedure of the proposed DQN and convex optimization algorithm in Algorithm 1.

Algorithm 1 DQN and convex optimization algorithm

```
Initialize the memory of experience replay ER
Randomly initialize the actor-network with parameter θ and the target-network with parameter θ̂, set θ̂ = θ
for episode=1, 2, 3, ... do
    Initialize s_0, p_m, f_m, h_m
    for t=1, 2, 3, ... do
        Randomly select a_t ∈ A
        Calculate the offloading ratio α through a_t
        Calculate f_m through (24)
        Obtain the reward r_t through calculating the total latency with α and f_m and obtain the state s_{t+1}
        Store the transition sample (s_t, a_t, R_{a_t}, s_{t+1}) in ER
        Randomly sample a mini-batch of transitions from ER and calculate L_t via (16) every T_l time slots
        Execute BP algorithm to update θ
        Reset θ̂ = θ every T_u time slots
    end for
end for
```

3.3 Some discussions on the system design and optimization

Besides the above works, one should note that there maybe exist some malicious vehicles which may overheat the confidential message from the data offloading. In this case, some privacy protection methods such as the encryption [36] and physical-layer secure schemes [37] should be used to enhance the security of the considered IoV networks. Moreover, some novel wireless techniques should be incorporated into the considered system, such as advanced offloading strategies [38], relaying techniques [39–41], and UAV [42, 43]. In further, some intelligent algorithms should be developed to allocate the system resources in a much more intelligent approach, such as deep learning [44–46], deep reinforcement learning [47] and federated learning [48].

4 Results and discussion

This section shows the performance of the proposed DQN and convex optimization algorithm in the vehicular MEC network from simulations. Channels used in our work obey Rayleigh flat fading, and the variance of AWGN is 0.01. The task sizes of users are $l_m = (100 + 5 \times m)$ Mb, and the transmit power of each user is randomly set to either 2 W or 3 W. Moreover, the calculating capability of users is set to 2×10^8 cycle/sec, and the number of required cycles per bit of data for computing is set as $C = 40$. All results given in this paper are the average of 5 experimental results.

As to the network structure, we implement both the target-network and actor-network of DQN through two hidden layers with 256 and 64 nodes and employ the BP algorithm as the updater. The values of T_l and T_u in the DQN are set to 50 and 100,

respectively. The size of experience replay is set to 20000, and the mini-batch size of sampling is set to 32.

Figure 3 plots the latency of the devised scheme versus the number of episodes, where the number of users is set to 3, the budget of users is set to 210, the total calculating capability of CAP is set to 5×10^8 cycle/sec, and the bandwidth of a wireless link is 1 MHz. To compare with the proposed DQN and convex optimization scheme, we plot the performance of two other schemes. One is the All-local scheme where users compute their tasks locally, and another is the ALL-CAP scheme where users offload all tasks to the CAP and obtain the calculating capability from CAP with the maximum budget. From this figure, we can see that the system latency of the proposed scheme gradually decreases when the episode varies from 0 to 20, and it converges to about 10 after 20 episodes. In contrast, the system latency of All-local scheme and ALL-CAP scheme remains unchanged at the level of 23 and 17.5, respectively. The fast convergence of the proposed scheme indicates that it can obtain an effective offloading decision and the calculating capability allocation. Moreover, the proposed scheme has the best performance among the three plotted schemes. Specifically, the system latency of the proposed scheme is about 56% and 10% lower than that of All-local scheme and ALL-CAP scheme. Obviously, the proposed scheme can not only converge rapidly but also outperform the other two schemes.

Figure 4 demonstrates the convergence of the proposed scheme with different numbers of users, where the total calculating capability at the server, the budget of users, and the bandwidth of a wireless link are set to 5×10^8 cycle/sec, 210, and 5 MHz, respectively. The figure shows that for the different numbers of users, the system latency of the devised scheme decreases in the first 30 episodes, and it converges to a low latency after 30 episodes. This result indicates that the proposed scheme can converge under various numbers of users. Moreover, the value of convergence increases with a larger M. This is because increasing the number of users causes more calculating tasks, which results in larger system latency. This further illustrates that the proposed scheme obtains a reasonable offloading decision and calculating capability allocation for different numbers of vehicles.

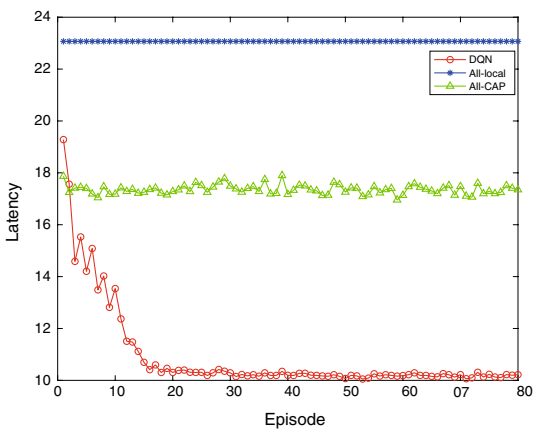

Fig. 3 Performance comparison of three scheme versus the number of episodes

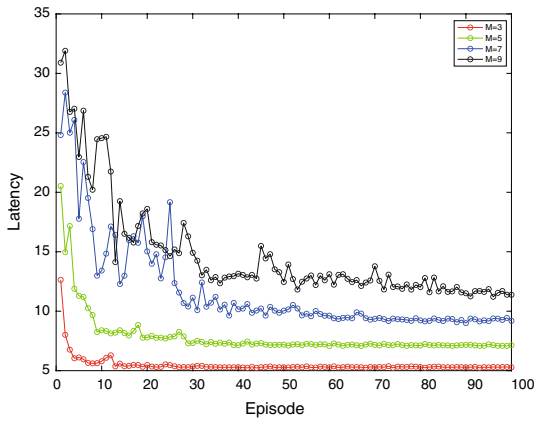

Fig. 4 Convergence of the proposed scheme with different number of users

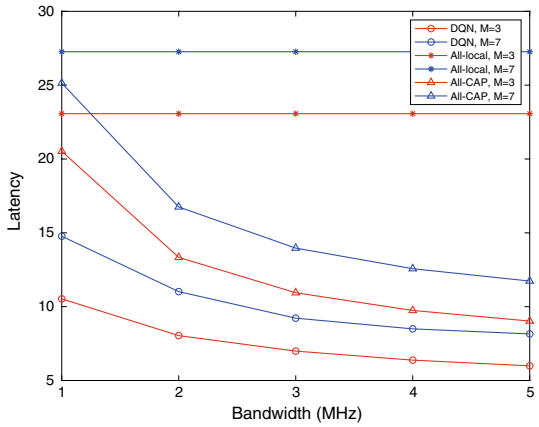

Fig. 5 Effect of the bandwidth on the system latency of the three schemes

Figure 5 shows the effect of wireless bandwidth on the system latency, where the total calculating capability at the server is 5×10^8 cycle/sec, the budget of users is set to 130, and the bandwidth varies from 1 MHz to 5 MHz. This figure shows that the system latency of the proposed scheme and ALL-CAP scheme drops sharply when the bandwidth varies from 1 MHz to 3 MHz and becomes steady when $W > 3$ MHz, while the system latency of All-local scheme remains unchanged with various values of bandwidth. The reason for this trend is that a larger bandwidth can reduce the transmission latency of the proposed scheme and ALL-CAP scheme, while the tasks of All-local scheme are not transmitted to the CAP. Moreover, the proposed scheme can obtain a lower latency for various values of bandwidth compared with the other two schemes. Specifically, when the bandwidth is 5 MHz, the latency of the proposed scheme is about 70% and 40% lower than that of All-local and ALL-CAP scheme. In further, for the three schemes, the system latency with $M = 3$ is always lower than that with $M = 7$. This is because a larger amount of tasks are produced due to the increasing number of users, which causes more communication and computation latency in the network. These results illustrate that the proposed scheme can outperform the other two schemes.

Figure 6 reveals the effect of the total calculating capability on the system latency of the proposed scheme with the variation of the total calculating capability of CAP under different bandwidths, where the number of users is set to 3, the budget of users is set to 210, and the total calculating capability at the server varies from 1×10^8 cycle/sec to 5×10^8 cycle/sec. This figure illustrates that the system latency of the devised scheme decreases when the total calculating capability at the CAP increases. This is because the CAP with a larger calculating capability can help users compute the tasks quickly, which leads to a reduction in the system latency. Moreover, with the variation of the total calculating capability from 1×10^8 cycle/sec to 5×10^8 cycle/sec, the performance improvement due to the enhanced calculating capability at the CAP becomes larger when the bandwidth increases. Specifically, the system latency drops by about 58% from $F = 1 \times 10^8$ cycle/sec to $F = 5 \times 10^8$ cycle/sec at $W = 5$ MHz, and it drops by 20% at $W = 1$ MHz. This is because the bandwidth affects the transmission latency and the total calculating capability affects the calculating latency at the CAP. These two types of latency both contribute to the system latency.

Figure 7 demonstrates the effect of the user budget on the system latency, where the total calculating capability at the CAP is set to 5×10^8 cycle/sec, the bandwidth is set to 5 MHz, and the budget of users varies from 70 to 150. This figure expresses that the system latency of the proposed scheme firstly decreases with the budget from 70 to 110, and then, it becomes steady when $U_m^{\max} \geq 110$, while that of All-local scheme is unchanged. This is because when the user budget is small, the calculating capability at the CAP allocated to the users is limited, which results in high calculating latency in the system. Moreover, the calculating capability allocated to each user decreases as the user number increases. This also verifies that the proposed scheme can make an effective offloading decision and calculating capability allocation compared with the All-local scheme.

Figure 8 illustrates the effect of the total CAP calculating capability and the user budget on the system latency, where the number of users is set to 3, the bandwidth is 1 MHz, the total calculating capability of the server varies from 1×10^8 cycle/sec to 5×10^8 cycle/sec, and the budget of users varies from 30 to 190. Observing this figure, we can see that the system latency of the proposed scheme is marginally affected

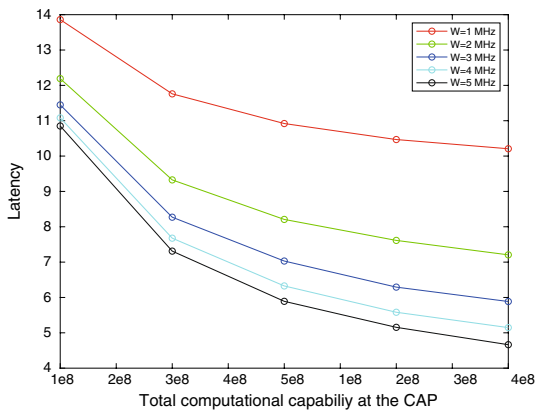

Fig. 6 Effect of the total calculating capability of CAP on the system latency of the proposed scheme

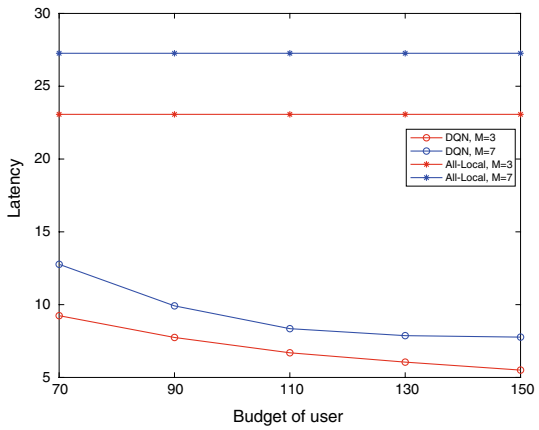

Fig. 7 Effect of budget on the system latency of the proposed scheme with $M = 3$ and $M = 7$

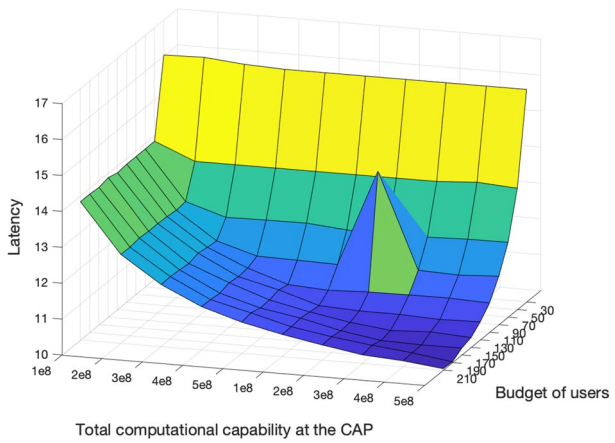

Fig. 8 Effect of the total CAP calculating capability and the user budget on the system latency

by the calculating capability at the CAP, when the user budget is small. The reason is that the budget limits the allocation of calculating capability to the vehicles, no matter how the total calculating capability changes. Similarly, when the total calculating capability at the server is small, the system latency is also marginally affected by the user budget, no matter how the budget changes. This is because the CAP does not have enough calculating capability to help the users compute the tasks. On the contrary, when the budget and total calculating capability at the CAP are both large, the system latency can be reduced to a small value. This is because when the CAP has enough calculating capability and users are rich enough, the allocation of calculating resources is no longer constrained, which makes a low system latency. All the above phenomena show that the system latency is limited by both the CAP calculating capability and the budget of users, and also indicate that the devised scheme in this paper has good performance in reducing system latency.

5 Conclusion

This article studied a vehicle MEC network, in which the CAP with limited calculating capability could receive part tasks from users to do the faster process, which can reduce system latency. We firstly formulated the optimization problem of latency by considering the limited calculating capability at the CAP and the budget of users. Then, we proposed a DQN and convex optimization algorithm to solve the problem. Simulations were finally conducted to show that the devised algorithm performs better than traditional methods, and it is robust to practical conditions of the vehicular MEC networks. As to future works, we consider that in the MEC networks, multiple CAPs can provide more options for users to do offloading, which can further reduce the calculating latency. Moreover, the computing of tasks and the transmission of tasks cause energy consumption, and the energy consumption is another important performance metric of the MEC networks in some scenarios. Therefore, we will consider multiple CAPs for the MEC networks and study the energy consumption in future works.

Abbreviations

MEC: Mobile edge computing; IoV: Internet of vehicle; QoS: Quality of service; DRL: Deep reinforcement learning; DQN: Deep Q-network; CAP: Calculating access point; DDPG: Deep deterministic policy gradient; UE: User equipment; AWGN: Additive white Gaussian noise..

Acknowledgements

This work was supported by the Key-Area Research and Development Program of Guangdong Province (No. 2019B090904014), by the International Science and Technology Cooperation Projects of Guangdong Province (No. 2020A0505100060), by the Natural Science Foundation of Guangdong Province (No. 2021A1515011392), and in part by the research program of Guangzhou University (No. YJ2021003).

Author Contributions

L.Z. designed the proposed framework and performed the simulations, W.Z. helped improve the optimization method, J.X. helped revise the manuscript in both the structure and grammar check, C.G. helped perform the simulations in this work, F.Z. helped enhance the design of the deep neural networks, C.F. helped explain the simulation results in this paper, and J.O. helped clarify the main contribution of this paper. J.X., C.G., and F.Z. are the corresponding authors of this paper. All authors read and approved the final manuscript.

Funding

This work was supported by the Key-Area Research and Development Program of Guangdong Province (No. 2019B090904014), by the International Science and Technology Cooperation Projects of Guangdong Province (No. 2020A0505100060), by the Natural Science Foundation of Guangdong Province (No. 2021A1515011392), and in part by the research program of Guangzhou University (No. YJ2021003).

Availability of data and materials

The authors state the data availability in this manuscript.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Author details

[1]School of Computer Science, Guangzhou University, Guangzhou, China. [2]Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China. [3]AI Sensing Technology, Chancheng District, Foshan, China.

Received: 17 January 2022 Accepted: 7 May 2022
Published online: 26 May 2022

References

1. M. Dai, Z. Zheng, S. Zhang, H. Wang, X. Lin, SAZD: a low computational load coded distributed computing framework for iot systems. IEEE Internet Things J. **7**(4), 3640–3649 (2020)
2. X.B. Zhai, L. Zheng, C.W. Tan, Energy-infeasibility tradeoff in cognitive radio networks: Price-driven spectrum access algorithms. IEEE J. Sel. Areas Commun. **32**(3), 528–538 (2014)
3. S. Tang, Dilated convolution based CSI feedback compression for massive MIMO systems. IEEE Trans. Veh. Technol. **71**(5), 211–216 (2022)
4. X. Hu, C. Zhong, Y. Zhu, X. Chen, Z. Zhang, Programmable metasurface-based multicast systems: Design and analysis. IEEE J. Sel. Areas Commun. **38**(8), 1763–1776 (2020)
5. X. Hu, Y. Zhang, X. Chen, Z. Zhang, Location information aided multiple intelligent reflecting surface systems. IEEE Trans. Commun. **68**(12), 7948–7962 (2020)
6. B. Li, Z. Na, B. Lin, Uav trajectory planning in harsh environment from a comprehensive energy efficiency perspective. IEEE Network, 1–7 (2022)
7. T. Li, C. Gao, L. Jiang, W. Pedrycz, J. Shen, Publicly verifiable privacy-preserving aggregation and its application in IoT. J. Netw. Comput. Appl. **126**, 39–44 (2019)
8. X. Hu, J. Wang, Statistical CSI based design for intelligent reflecting surface assisted MISO systems. Sci. China Inf. Sci. **63**(12), 222303 (2020)
9. X. Lai, Y. Deng, G.K. Karagiannidis, A. Nallanathan, Secure mobile edge computing networks in the presence of multiple eavesdroppers. IEEE Trans. Commun. **70**(1), 500–513 (2022)
10. N. Zhenyu, L. Bowen, L. Xin, W. Jun, Z. Mengshu, L. Yue, M. Beihang, UAV-based wide-area internet of things: An integrated deployment architecture. IEEE Netw. **35**(5), 122–128 (2021)
11. L. Chen, Intelligent ubiquitous computing for future UAV-enabled MEC network systems. Clust. Comput. **2021**(25), 1–10 (2021)
12. J. Zhang, Y. Zhang, Z. Zhang, Robust design for intelligent reflecting surfaces assisted MISO systems. IEEE Commun. Lett. **24**(10), 2353–2357 (2020)
13. W. Lin, T. Yu, C. Gao, F. Liu, T. Li, S. Fong, Y. Wang, A hardware-aware CPU power measurement based on the power-exponent function model for cloud servers. Inf. Sci. **547**, 1045–1065 (2021)
14. L. Hu, H. Yan, L. Li, Z. Pan, X. Liu, Z. Zhang, MHAT: an efficient model-heterogenous aggregation training scheme for federated learning. Inf. Sci. **560**, 493–503 (2021)
15. R. Zhao, M. Tang, Profit maximization in cache-aided intelligent computing networks. Phys. Commun. **99**, 1–10 (2022)
16. J. Lu, M. Tang, Performance analysis for IRS-assisted MEC networks with unit selection. Phys. Commun. **99**, 1–10 (2022)
17. Y. Wu, C. Gao, Intelligent task offloading for vehicular edge computing with imperfect CSI: A deep reinforcement approach. Phys. Commun. **99**, 1–10 (2022)
18. X. Liu, C. Sun, M. Zhou, C. Wu, B. Peng, P. Li, Reinforcement learning-based multislot double-threshold spectrum sensing with bayesian fusion for industrial big spectrum data. IEEE Trans. Ind. Inf. **17**(5), 3391–3400 (2021). https://doi.org/10.1109/TII.2020.2987421
19. Lai, X.: Outdated access point selection for mobile edge computing with cochannel interference. IEEE Trans. Veh. Technol. 99, 1–12 (2021)
20. Lu, J.: Analytical offloading design for mobile edge computing based smart internet of vehicle. EURASIP J. Adv. Signal Process. 99, 1–10 (2022)
21. S. Tang, W. Zhou, L. Chen, L. Lai et al., Battery-constrained federated edge learning in UAV-enabled IoT for B5G/6G networks. Phys. Commun. **47**, 101381 (2021)
22. Y. Liao, X. Qiao, Q. Yu, Q. Liu, Intelligent dynamic service pricing strategy for multi-user vehicle-aided MEC networks. Future Gener. Comput. Syst. **114**, 15–22 (2021)
23. Z. Na, Y. Liu, J. Shi, C. Liu, Z. Gao, Uav-supported clustered noma for 6g-enabled internet of things: Trajectory planning and resource allocation. IEEE Internet Things J. **8**(20), 15041–15048 (2021). https://doi.org/10.1109/JIOT.2020.3004432
24. S. Tang, L. Chen, Computational intelligence and deep learning for next-generation edge-enabled industrial IoT. IEEE Trans. Netw. Sci. Eng. **99**, 1–12 (2022)
25. X. Liu, X. Zhang, NOMA-based resource allocation for cluster-based cognitive industrial internet of things. IEEE Trans. Ind. Informat. **16**(8), 5379–5388 (2020). https://doi.org/10.1109/TII.2019.2947435
26. F. Li, K.-Y. Lam, X. Liu, J. Wang, K. Zhao, L. Wang, Joint pricing and power allocation for multibeam satellite systems with dynamic game model. IEEE Trans. Veh. Technol. **67**(3), 2398–2408 (2018). https://doi.org/10.1109/TVT.2017.2771770
27. L. He, K. He, Towards optimally efficient search with deep learning for large-scale MIMO systems. IEEE Trans. Commun. **70**(5), 3157–3168 (2022)
28. S. Tang, X. Lei, Collaborative Cache-Aided Relaying Networks: Performance Evaluation and System Optimization. IEEE J. Sel. Areas Commun. 1–12 (2022)
29. K. He, Y. Deng, Efficient memory-bounded optimal detection for GSM-MIMO systems. IEEE Trans. Commun. **99**, 1–12 (2022)
30. Tiwari, M., Maity, I., Misra, S.: Loan: Latency-aware task offloading in association-free social fog-iov networks. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 01–06 (2021). https://doi.org/10.1109/GLOBECOM46510.2021.9685399
31. Z. Wenqi, C. Lunyuan, T. Shunpu, L. Lijia et al., Offloading strategy with PSO for mobile edge computing based on cache mechanism. Clust. Comput. **2021**, 1572–7543 (2021)
32. L. Zhang, C. Gao, Deep reinforcement learning based IRS-assited mobile edge computing under physical-layer security. Phys. Commun. **99**, 1–10 (2022)
33. L. Chen, Physical-layer security on mobile edge computing for emerging cyber physical systems. Comput. Commun. **99**, 1–12 (2022)

34. Y. Guo, S. Lai, Distributed machine learning for multiuser mobile edge computing systems. IEEE J. Sel. Top. Signal Process. **16**(3), 460–473 (2021)
35. R. Zhao, M. Tang, Impact of direct links on intelligent reflect surface-aided MEC networks. Phys. Commun. **99**, 1–10 (2022)
36. H. Yan, L. Hu, X. Xiang, Z. Liu, X. Yuan, PPCL: privacy-preserving collaborative learning for mitigating indirect information leakage. Inf. Sci. **548**, 423–437 (2021)
37. F. Shi, J. Xia, Z. Na, X. Liu, Y. Ding, Z. Wang, Secure probabilistic caching in random multi-user multi-uav relay networks. Phys. Commun. **32**, 31–40 (2019)
38. S. Zhu, W. Xu, L. Fan, K. Wang, G.K. Karagiannidis, A novel cross entropy approach for offloading learning in mobile edge computing. IEEE Wirel. Commun. Lett. **9**(3), 402–405 (2020)
39. M. Fang, D. Li, H. Zhang, L. Fan, I. Trigui, Performance analysis of short-packet communications with incremental relaying. Comput. Commun. **177**, 51–56 (2021)
40. H. Huang, J. Xia, X. Liu, Z. Na, Q. Yang, H. Chen, J. Zhao, Switch-and-stay combining for energy harvesting relaying systems. Phys. Commun. **28**, 28–34 (2018)
41. P.S. Bouzinis, P.D. Diamantoulakis, L. Fan, G.K. Karagiannidis, Pareto-optimal resource allocation in decentralized wireless powered networks. IEEE Trans. Commun. **69**(2), 1007–1020 (2021)
42. X. Lin, J. Xia, Z. Wang, Probabilistic caching placement in uav-assisted heterogeneous wireless networks. Phys. Commun. **33**, 54–61 (2019)
43. W. Zhou, D. Deng, J. Xia, Z. Shao, The precoder design with covariance feedback for simultaneous information and energy transmission systems. Wirel. Commun. Mob. Comput. **2018**, 8472186–1847218617 (2018)
44. K. He, L. He, L. Fan, Y. Deng, G.K. Karagiannidis, A. Nallanathan, Learning-based signal detection for MIMO systems with unknown noise statistics. IEEE Trans. Commun. **69**(5), 3025–3038 (2021)
45. W. Chen, J. Li, Z. Huang, C. Gao, S. Yiu, Z.L. Jiang, Lattice-based unidirectional infinite-use proxy re-signatures with private re-signature key. J. Comput. Syst. Sci. **120**, 137–148 (2021)
46. T. Huang, Q. Zhang, J. Liu, R. Hou, X. Wang, Y. Li, Adversarial attacks on deep-learning-based SAR image target recognition. J. Netw. Comput. Appl. **162**, 102632 (2020)
47. M. Kanghua, T. Weixuan, L. Jin, Y. Xu, Attacking deep reinforcement learning with decoupled adversarial policy. IEEE Trans. Dependable Secur. Comput. **18**(5), 2438–2455 (2022)
48. Z. Zhao, X. Lei, G.K. Karagiannidis, A. Nallanathan, System optimization of federated learning networks with a constrained latency. IEEE Trans. Veh. Technol. **71**(1), 1095–1100 (2022)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com