DQN Based Mobile Edge Computing for Smart Internet of Vehicle

Lianhong Zhang (✉ 2112106071@e.gzhu.edu.cn)

Guangzhou University <https://orcid.org/0000-0001-5351-1304>

Wenqi Zhou

Guangzhou University

Junjuan Xia

Guangzhou University

Chongzhi Gao

Guangzhou University

Fusheng Zhu

New Generation University College

Research Article

Keywords: Internet of vehicle, mobile edge computing, budget, offloading strategy, latency

Posted Date: January 25th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1260715/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

DQN Based Mobile Edge Computing for Smart Internet of Vehicle

Lianhong Zhang^{1*}, Wenqi Zhou¹, Junjuan Xia¹, Chongzhi Gao¹ and Fusheng Zhu²

*Correspondence:

2112106071@e.gzhu.edu.cn

¹School of Computer Science,
Guangzhou University, 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 computational tasks from the vehicular users. For the MEC networks, most of 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 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 computational capability of the server to multiple users. Simulations are finally presented to demonstrate that the proposed schemes outperforms the conventional ones. In particular, the proposed scheme can effective reduce the system latency 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, 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 [1]. In the IoV systems, mobile vehicles can collect data and exert some intensive computational tasks to make intelligent decision [2,3]. However, due to the lack of computational capability at mobile vehicles, local computing may cause a significant latency and even lead to some serious consequences, e.g, traffic accidents [4]. By utilizing the remote computational resources at the cloud server, the computational latency can be reduced effectively, at the cost of an increased communication latency [5–8]. In particular, the communication latency may become the bottleneck of the system performance when the wireless channels are vulnerable. Moreover, massive requests arising from the network edge impose a heavy burden to the cloud server, which further deteriorates the quality of experience (QoE) of users.

To address the above issues of cloud computing, a novel communication and computation paradigm named mobile edge computing (MEC) was proposed. By deploying computational access points (CAPs) at the network edge, users can offload their computational tasks to the nearby CAPs through reasonable task partition

and offloading, in order to achieve a low latency and energy consumption [9–11]. In this direction, the authors in [12–16] investigated a multiuser multi-CAP MEC network, and proposed a deep Q-network (DQN) based offloading strategy for the task offloading. Moreover, the system cost was studied in [17–20] in terms of a combination of latency and energy consumption, where a joint optimization framework of offloading decision and resource allocation was proposed to enhance the network performance. In further, a deep deterministic policy gradient (DDPG) was proposed to solve the offloading strategy design in the MEC networks [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 equipments (UEs).

From the above literature review, we can find that most of the existing works tried to optimize the system performance of MEC networks through offloading strategy design and resource allocation. To the best of our knowledge, seldom works consider the pricing from the server and take into account the 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 allocated computational capability from the CAPs to the users. In addition, the budger 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 computational tasks have to be computed at local. 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 computational tasks from the vehicular users. For the MEC networks, most of 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 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 computational capability of the server to multiple users. Simulations are finally presented to demonstrate that the proposed schemes outperforms the conventional ones. In particular, the proposed scheme can effective reduce the system latency up to 56% compared to the conventional schemes.

The rest of this paper is organized as follows. After the Introduction, we will discuss the system model of MEC based IoV network and then present the optimization problem formulation in Sec. II. After that, we give the DQN and convex optimization algorithm based method to solve the optimization problem in Sec. III. Then, we provide some simulations and discussions in Sec. IV and finally make some conclusions in Sec. V.

2 Methods/experimental

As shown in Fig. 1, we consider 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 computational 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 task and the computational capability. In the following subsection, we will detail 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 computational latency is written as

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

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

2.2 Offloading Model

When some parts of the tasks are offloaded to the CAP for computing, the users should transmit the offloaded tasks through wireless link, and the data transmission rate can be given as

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

where W is the bandwidth of the wireless link between user u_m and CAP, P_m is the transmission power of user u_m , $h_m \sim \mathcal{CN}(0, \beta)$ is the channel parameter of the wireless link, and σ^2 is the variance of additive white Gaussian noise (AWGN) at the CAP. From R_m , the transmission latency can be given by

$$T_m^{tran} = \frac{l_m \alpha_m}{R_m}. \quad (3)$$

After receiving the tasks from user u_m , the CAP begins to compute the tasks, and the computational latency at the CAP can be given as

$$T_m^{CAP} = \frac{l_m \alpha_m C}{f_m}. \quad (4)$$

where f_m is the computational capability that the user u_m buys from the CAP, and it should satisfy the constraint $\sum_{m=1}^M f_m \leq F$, in which F is the total computational capability of the CAP.

Then, we can get the offloading latency including the transmission latency and the computational latency at the CAP, given by

$$T_m^{offl} = T_m^{tran} + T_m^{CAP}. \quad (5)$$

Since the local computation and the offloading are two operations that can be executed in parallel, the total computational latency of task of user u_m can be given as

$$T_m^{total} = \max\{T_m^{local}, T_m^{offl}\}. \quad (6)$$

For the whole vehicular MEC network, the tasks from each users can be processed parallely. Therefor, the system latency can be defined as the finish time of the tasks from all users, given by

$$T^{total} = \max\{T_1^{total}, T_2^{total}, T_3^{total}, \dots, T_M^{total}\}. \quad (7)$$

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 fee and computational fee. Specifically, the basic service fee is based on the size of offloading tasks, and the computational fee is based on the computational capability that the CAP allocates to the users. Hence, the payment of user u_m for offloading can be given as

$$U_m = \eta_l l_m \alpha_m + \eta_f f_m, \quad (8)$$

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

$$U_m \leq U_m^{max}, \quad (9)$$

Table 1 Symbol notations.

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

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

2.4 Problem Formulation

In the practical vehicular MEC network, 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

$$\mathbf{P1} : \min_{\{\alpha_m, f_m\}} T^{total} \quad (10a)$$

$$s.t. \quad C_1 : \alpha_m \in [0, 1], \forall m \in [1, M], \quad (10b)$$

$$C_2 : \sum_{m=1}^M f_m \leq F, \quad (10c)$$

$$C_3 : U_m \leq U_m^{max}, \forall m \in [1, M], \quad (10d)$$

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 computational capability of the CAP allocates to user u_m should not exceed the total computational capability. The last constraint C_3 denotes that users' payment of offloading should meet the budget constraint. From (10a), we can optimize the offloading ratio and computational capability to minimize the system latency while meeting the constraint of the budget. However, the optimization problem is complicated and hard to be solved by conventional convex optimization method. Therefore, we will propose a DQN and convex optimization algorithm to solve the problem. The notations of this section we have used are summarized in Table 1.

3 DQN and Convex Optimization Algorithm

In this section, we propose a DQN and convex optimization algorithm to solve the problem in **P1**. In the following subsections, we firstly describe how to obtain the offloading decision through the DQN, and give the process of the resource allocation through the convex optimization method.

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 computational capability allocation. As shown in Fig. 2, the proposed algorithm is composed of DQN based method and convex optimization method. Specifically, we first adopt 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 computational capability.

The problem of offloading decision can be modeled by Markov decision process (MDP). In MDP, the agent firstly observes the state $s_t \in \mathcal{S}$ from the environment at time slot t , and makes an action a_t according to policy π . Then, the agent transfers the state s_t to state s_{t+1} by making action a_t in the environment and get a reward r_t . Specifically, we define the state space as $\mathcal{S} = \{\alpha\}$, where $\alpha = \{\alpha_1(t), \alpha_2(t), \alpha_3(t), \dots, \alpha_M(t)\}$ is the offloading ratio of users at the time slot t , and the action space is $\mathcal{A} = \{\rho_1, \rho_2, \rho_3, \dots, \rho_m, \dots, \rho_M, \rho_1^*, \rho_2^*, \rho_3^*, \dots, \rho_m^*, \dots, \rho_M^*\}$, 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, given by

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

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 action a acts on state s . According to Q function, the best policy can be given as

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

and the agent observes the state s from the environment to choose the action according to the best policy π^* , given by

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

Based on the above process, we adopt deep Q-Network (DQN) which uses deep neural network to approximate the optimal Q function. There are two neural networks in DQN, including the actor-network and the target-network. The actor-network is used to predict the action $a_t \in A$ by inputting the state $s_t \in \mathcal{S}$. Without loss of generality, we use $Q(s, a)$ to denote the $Q_{\pi^*}(s, a)$. In order to avoid the offloading optimization problem fallings into local optimal value, the action is given by the ϵ -greedy policy, which can be given as

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

where $\boldsymbol{\theta}$ is the parameter of the actor-network. To help approximate the Q function, we adopt the temporal-difference (TD) approach in DQN, which can be given as

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

In general, in order to calculate 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 can be given as.

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

where $\hat{\boldsymbol{\theta}}$ is the parameter 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 a experience replay (ER) and a mini-batch sampling. A transition (s_t, a_t, r_t, s_{t+1}) is stored into ER at every 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 giving the offloading decision α_m by the DQN-network, the problem **P1** can be transformed into

$$\mathbf{P2} : \min_{\{f_m\}} T^{total} \quad (17a)$$

$$s.t. \quad C_1 : \sum_{m=1}^M f_m \leq F, \quad (17b)$$

$$C_2 : U_m \leq U_m^{max}, \forall m \in [1, M]. \quad (17c)$$

From **P2**, we can observe that the minimization problem is affected by the total computational capability of the CAP and the budget of the user. To simplify the problem, we firstly relax the constraint C_2 and transform **P2** into a convex problem,

given by

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

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

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

where $\lambda > 0$ is Lagrange multiplier. From (19), we set the first partial derivatives of $\mathcal{L}(f_m, \lambda)$ with f_m and λ to zero respectively, given by

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

By combing the above equations, we can obtain the optimal solution of **P3** as

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

$$f_m = \frac{l_m \alpha_m F}{\sum_{m=1}^M \sqrt{l_m \alpha_m}}. \quad (22)$$

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

$$0 \leq f_m \leq \frac{U_m^{max} - \eta_l l_m \alpha_m}{\eta_f}. \quad (23)$$

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

$$f_m = \begin{cases} \frac{l_m \alpha_m F}{\sum_{m=1}^M \sqrt{l_m \alpha_m}}, & \text{If } \frac{l_m \alpha_m F}{\sum_{m=1}^M \sqrt{l_m \alpha_m}} \leq \frac{U_m^{max} - \eta_l l_m \alpha_m}{\eta_f}, \\ \frac{U_m^{max} - \eta_l l_m \alpha_m}{\eta_f}, & \text{Else.} \end{cases} \quad (24)$$

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

```

1: Initialize the memory of experience replay  $ER$ 
2: Randomly initialize the actor-network with parameter  $\theta$  and the target-network with parameter  $\hat{\theta}$ ,
   set  $\hat{\theta} = \theta$ 
3: for episode=1, 2, 3, ... do
4:   Initialize  $s_0, p_m, f_m, h_m$ 
5:   for  $t=1, 2, 3, \dots$  do
6:     Randomly select  $a_t \in A$ 
7:     Calculate the offloading ratio  $\alpha$  through  $a_t$ 
8:     Calculate  $f_m$  through (24)
9:     Obtain the reward  $r_t$  through calculating the total latency with  $\alpha$  and  $f_m$  and obtain the
       state  $s_{t+1}$ 
10:    Store the transition sample  $(s_t, a_t, R_{a_t}, s_{t+1})$  in  $ER$ 
11:    Randomly sample a mini-batch of transitions from  $ER$  and calculate the  $L_t$  via 16 every
        $T_l$  time slots
12:    Execute BP algorithm to update  $\theta$ 
13:    Reset  $\hat{\theta} = \theta$  every  $T_u$  time slots
14:   end for
15: end for

```

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

4 Results and discussions

In this section, we conduct the simulations to illustrate the performance of the proposed DQN and convex optimization algorithm in the vehicular MEC network. All channels in the network experience Rayleigh flat fading, and the variance of AWGN is set to 0.01. Without loss of generality, the task sizes of users are set to $l_m = (100 + 5 \times m)$ Mb and the transmission power of user u_m is randomly set to either 2 W or 3 W. Moreover, the computational capability of users is set to 2×10^8 cycle/sec, and the required cycles of per bit data for computing is set as $C = 40$. All results given in this paper are the average of 5 experimental results.

As for 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.

Fig. 3 shows the latency of the proposed 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

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

computational capability of CAP is set to 5×10^8 cycle/sec, and the bandwidth of wireless link is set to 1 MHz. To compare with the proposed DQN and convex optimization scheme, we plot the performance of All-local scheme, where users compute their tasks locally, and the performance of the All-cap scheme where users offload all tasks to the CAP and obtain the computational capability from CAP with the maximum budget. From this figure, we can see that the system latency of the proposed scheme gradually decreases when 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 11, respectively. The fast convergence of the proposed scheme indicates that it can obtain an effective offloading decision and the computational capability allocation. Moreover, the proposed scheme outperforms All-local and All-cap. 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.

Fig. 4 demonstrates the convergence of the proposed scheme with different number of users, where the total computational capability of the CAP, the budget of users, and the bandwidth of wireless link are set to 5×10^8 cycle/sec, 210, and 5 MHz, respectively. From the figure, we can observe that for various numbers of users, the system latency of the proposed 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 number of users. Moreover, the value of convergence increases with a larger M . This is because that increasing the number of users causes more computational tasks, which result in a larger system latency. This further illustrates that the proposed scheme obtains a reasonable offloading decision and computational capability allocation for various numbers of users.

Fig. 5 shows the effect of wireless bandwidth on the system latency, where the total computational capability of the CAP is set to 5×10^8 cycle/sec, the budget of users is set to 130, and the bandwidth varies from 1 MHz to 5 MHz. We can observe from this figure that the system latency of the proposed scheme and All-cap scheme drops

Figure 5 Effect of the bandwidth on the system latency with three schemes.

Figure 6 Effect of the total computational capability of CAP on the system latency of the proposed scheme with different bandwidth.

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. This is because that a larger bandwidth can help 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 proposed scheme is about 70% and 40% lower than that of All-local and All-cap scheme. This illustrates that the proposed scheme can outperform the other two schemes by optimizing the offloading decision and computational capability allocation.

In further, for the three schemes, the system latency with $M = 3$ always lower than that of $M = 7$. This is because that a larger amount of tasks are produced due to the increasing number of users, which causes more communication and computation latency in the network.

Fig. 6 reveals the effect of the total computational capability on the system latency of the proposed scheme with the variation of the total computational 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 computational capability of the CAP varies from 1×10^8 cycle/sec to 5×10^8 cycle/sec. We can find from this figure that the system latency of the proposed scheme decreases when the total computational capability at the CAP increases. This is because that the CAP with a larger computational capability can help users compute the tasks quickly, which leads to the reduction in the system latency. Moreover, with the variation of the total computational capability from 1×10^8 cycle/sec to 5×10^8 cycle/sec, the performance improvement due to the enhanced computational 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 that the bandwidth affects the transmission latency and the total computational capability affects the computational latency at the CAP. These two latency both contribute to the system latency.

Fig. 7 demonstrates the effect of the user budget on the system latency, where the total computational 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. From this figure, we can observe that the system latency of the proposed scheme firstly decreases with the budget from 70 to 110, and then it becomes steady when $110 \leq U_m^{max}$, while that of All-local scheme is unchanged. This is because that when the user budget is small, the computational capability at the CAP allocated to the users is limited, which results in a computational latency in the system. This result shows that the computational 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 computational capability allocation compared with the All-local.

Fig. 8 illustrates the effect of the total CAP computational capability and the user budget on the system latency, where the number of users is set to 3, the bandwidth is

Figure 8 Effect of the of the total CAP computational capability and the user budget on the system latency.

set to 1 MHz, the total computational capability of CAP varies from 1×10^8 cycle/sec to 5×10^8 cycle/sec, and the budget of users varies from 30 to 190. Observing from this figure, we can see that the system latency of the proposed scheme is marginally affected by the computational capability at the CAP, when the user budget is small. The reason is that the budget limits the allocation of computational capability to the users, no matter how the total computational capability changes.

Similarly, when the total computational capability of the CAP 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 computational capability to help the users compute the tasks. On the contrary, when the budget and total computational capability at the CAP are both large, the system latency stabilizes at a small value. This is because that when the CAP has enough computational capability and users are rich enough, the allocation of computational 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 computational capability and the budget of users, and also indicate that the proposed scheme in this paper is effective in reducing the system latency.

5 Conclusion

In this paper, we studied a vehicle MEC network, where the CAP with limited computational capability can help users compute the tasks to reduce the system latency. We firstly formulated the optimization problem of latency, by considering the limited computational capability at the CAP and the budget of users. And then, we proposed a DQN and convex optimization algorithm to solve the problem. Simulations were finally conducted to show that the proposed algorithm has better performance than conventional method, and it is robust at any condition of the vehicular MEC.

Abbreviations

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

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The authors state the data availability in this manuscript.

Competing interests

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

Funding

This work was supported by the Key-Area Research and Development Program of Guangdong Province (No.2018B010124001), by the International Science and Technology Cooperation Projects of Guangdong Province (No. 2020A0505100060), by the National Natural Science Foundation of China (No. 61471229), by the Key Field Projects of Colleges and Universities of Guangdong Province (No. 2019KTSCX040), by the Natural Science Foundation of Guangdong Province (No. 2021A1515011392), and in part by the research program of Guangzhou University (No. YK2020008/YJ2021003).

Authors' contributions

L. Zhang designed the proposed framework and performed the simulations, W. Zhou helped improve the optimization method, J. Xia helped revise the manuscript in both the structure and grammar check, C. Gao was responsible for the check of the full paper and simulations and F. Zhu helped enhance the design of the deep neural networks. F. Zhu is the corresponding author of this paper. All authors read and approved the final manuscript.

Acknowledgements

This work was supported by the Key-Area Research and Development Program of Guangdong Province (No.2018B010124001), by the International Science and Technology Cooperation Projects of Guangdong Province (No. 2020A0505100060), by the National Natural Science Foundation of China (No. 61471229), by the Key Field Projects of Colleges and Universities of Guangdong Province (No. 2019KTSCX040), by the Natural Science Foundation of Guangdong Province (No. 2021A1515011392), and in part by the research program of Guangzhou University (No. YK2020008/YJ2021003).

Authors' information

L. Zhang, W. Zhou, J. Xia, and C. Gao are with the School of Computer Science, Guangzhou University, Guangzhou, China, (e-mail: {2112106071, 2112006056}@e.gzhu.edu.cn. { xiajunjuan, czgao}@gzhu.edu.cn). F. Zhu is with Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China (e-mail: zhufusheng@gdcni.cn).

Author details

¹School of Computer Science, Guangzhou University, Guangzhou, China. ²Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China.

References

1. Khoukhi, L., Xiong, H., Kumari, S., Puech, N.: The internet of vehicles and smart cities. *Ann. des Télécommunications* **76**(9-10), 545–546 (2021)
2. Huang, S., Gui, J., Wang, T., Li, X.: Joint mobile vehicle-uav scheme for secure data collection in a smart city. *Ann. des Télécommunications* **76**(9-10), 559–580 (2021)
3. Mlika, Z., Cherkaoui, S.: Network slicing for vehicular communications: a multi-agent deep reinforcement learning approach. *Ann. des Télécommunications* **76**(9-10), 665–683 (2021). doi:10.1007/s12243-021-00872-w
4. Anwar, M.R., Wang, S., Akram, M.F., Raza, S., Mahmood, S.: 5g-enabled MEC: A distributed traffic steering for seamless service migration of internet of vehicles. *IEEE Internet Things J.* **9**(1), 648–661 (2022)
5. Li, Y., Zhou, R., Liang, F., Ding, H., Ma, G., Wang, H.: Application research of cloud computing in electricity marketing field measurement remote acquisition system. *Mechatron. Syst. Control.* **50**(1) (2022)
6. Rui, L., Yang, S., Gao, Z., Li, W., Qiu, X., Meng, L.: Smart network maintenance in edge cloud computing environment: An allocation mechanism based on comprehensive reputation and regional prediction model. *J. Netw. Comput. Appl.* **198**, 103298 (2022)
7. Saha, S., Habib, M.A., Adhikary, T., Razzaque, M.A., Rahman, M.M., Altaf, M., Hassan, M.M.: Quality-of-experience-aware incentive mechanism for workers in mobile device cloud. *IEEE Access* **9**, 95162–95179 (2021)
8. Sharghivand, N., Derakhshan, F., Siasi, N.: A comprehensive survey on auction mechanism design for cloud/edge resource management and pricing. *IEEE Access* **9**, 126502–126529 (2021)
9. Wan, S., Wisniewski, R., Alexandropoulos, G.C., Gu, Z., Siano, P.: Special issue on optimization of cross-layer collaborative resource allocation for mobile edge computing, caching and communication. *Comput. Commun.* **181**, 472–473 (2022)
10. Mu, R., Gong, B., Ning, Z., Zhang, J., Cao, Y., Li, Z., Wang, W., Wang, X.: An identity privacy scheme for blockchain-based on edge computing. *Concurr. Comput. Pract. Exp.* **34**(1) (2022)

11. Yang, W., Thapliyal, H.: Approximate adiabatic logic for low-power and secure edge computing. *IEEE Consumer Electron. Mag.* **11**(1), 88–94 (2022)
12. Li, C., Xia, J., Liu, F., Li, D., Fan, L., Karagiannis, G.K., Nallanathan, A.: Dynamic offloading for multiuser multi-cap MEC networks: A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* **70**(3), 2922–2927 (2021)
13. Wu, Y., Dinh, T.Q., Fu, Y., Lin, C., Quek, T.Q.S.: A hybrid DQN and optimization approach for strategy and resource allocation in MEC networks. *IEEE Trans. Wirel. Commun.* **20**(7), 4282–4295 (2021)
14. Moon, J., Jeong, J.: Smart manufacturing scheduling system: DQN based on cooperative edge computing. In: Lee, S., Choo, H., Ismail, R. (eds.) 15th International Conference on Ubiquitous Information Management and Communication, IMCOM 2021, Seoul, South Korea, January 4–6, 2021, pp. 1–8. IEEE, ??? (2021)
15. Liu, H., Ma, Y., Chen, P., Xia, Y., Ma, Y., Zheng, W., Li, X.: Scheduling multi-workflows over edge computing resources with time-varying performance, A novel probability-mass function and dqn-based approach. In: Ku, W., Kanemasa, Y., Serhani, M.A., Zhang, L. (eds.) Web Services - ICWS 2020 - 27th International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, September 18–20, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12406, pp. 197–209. Springer, ??? (2020)
16. Xu, F., Yang, F., Qiu, C., Zhao, C., Li, B.: DQN aided edge computing in satellite-terrestrial network. In: Jia, M., Guo, Q., Meng, W. (eds.) Wireless and Satellite Systems - 10th EAI International Conference, WiSATS 2019, Harbin, China, January 12–13, 2019, Proceedings, Part I. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 280, pp. 117–127. Springer, ??? (2019)
17. Zhao, R., Wang, X., Xia, J., Fan, L.: Deep reinforcement learning based mobile edge computing for intelligent internet of things. *Phys. Commun.* **43**, 101184 (2020)
18. Guo, Y., Zhao, Z., He, K., Lai, S., Xia, J., Fan, L.: Efficient and flexible management for industrial internet of things: A federated learning approach. *Comput. Networks* **192**, 108122 (2021)
19. Guo, Y., Zhao, Z., Zhao, R., Lai, S., Dan, Z., Xia, J., Fan, L.: Intelligent offloading strategy design for relaying mobile edge computing networks. *IEEE Access* **8**, 35127–35135 (2020)
20. Tang, S., Chen, L., He, K., Xia, J., Fan, L., Nallanathan, A.: Computational intelligence and deep learning for next-generation edge-enabled industrial iot. *CoRR abs/2110.14937* (2021)
21. Tang, S., Zhou, W., Chen, L., Lai, L., Xia, J., Fan, L.: Battery-constrained federated edge learning in uav-enabled iot for B5G/6G networks. *Phys. Commun.* **47**, 101381 (2021)
22. Liao, Y., Qiao, X., Yu, Q., Liu, Q.: Intelligent dynamic service pricing strategy for multi-user vehicle-aided MEC networks. *Future Gener. Comput. Syst.* **114**, 15–22 (2021)