Multi-Vehicle Intelligent Collaborative Computing Strategy for Internet of Vehicles

Yaping Cui, Lijuan Du, Peng He, Dapeng Wu, Ruyan Wang.
School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing, China.
Advanced Network and Intelligent Connection Technology Key Laboratory of
Chongqing Education Commission of China, Chongqing, China.
Chongqing Key Laboratory of Ubiquitous Sensing and Networking, Chongqing, China.
Email: cuiyp@cqupt.edu.cn, dulijuan0706@163.com, hp6500@126.com,
wudp@cqupt.edu.cn, wangry@cqupt.edu.cn.

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771732

***Abstract*—The computation-intensive applications pose unprecedented demands on the Internet of Vehicles (IoVs). How to address the delay constraint to execute the computation tasks effectively becomes a significant issue for this scenario. Compared with remote cloud, edge servers reduce the delay by being deployed close to vehicles. However, most edge servers are connected to fixed access points, which leads to the inflexible edge computing architecture. Considering the dynamics of vehicles' location and service request, it is a promising paradigm that multi-vehicle compute the task collaboratively by utilizing the vehicles' available computing resources. In this paper, by jointly considering the local execution, V2V offloading, and multi-vehicle collaboration, we determine the optimal task partition ratio after the cooperative vehicles are selected. Then, double deep Q-network (DDQN) is used to take the optimal dual actions. Finally, we develop a multi-vehicle intelligent collaborative computing strategy (MV-ICCS) to minimize the total system delay. Simulation results show the advantage of the proposed strategy and evaluate the system performance.**

***Index Terms*—Internet of Vehicles (IoVs), Collaborative Computing, Double Deep Q-network (DDQN), Task Offloading.**

I. INTRODUCTION

With the continuous integration and breakthrough of new technologies, such as artificial intelligence (AI) and fifth-generation (5G) cellular communication, intelligent transportation has been brought to the stage of large-scale application, which promotes the development of computation-intensive applications such as autonomous driving, virtual reality and augmented reality [1]. Due to the limited computing resources of user terminals, there is a growing trend to offload the users' computation-intensive tasks onto mobile edge computing (MEC) servers.

Mobile edge computing, as an emerging paradigm to enhance the computation capacity at the edge of cellular networks, has attracted widespread attentions from academia and industry [2]. Nevertheless, traditional edge servers are connected to the fixed access points, such as base stations (BSs) and road side units (RSUs), which leads to the inflexible edge computing architecture. Moreover, the dynamics of vehicles' locations and service requests leads to a greater relative speed between the vehicle and the BS/RSU, it shortens the V2I link duration. On the contrary, the relative speed of two vehicles with same direction is small, there may be a longer connection time between them [3]. Therefore, we can offload computation to vehicles with available computing resources through V2V communication.

Collaborative computing by utilizing other vehicles' available computing resources is a promising solution to solve the explosive growth of data traffic and the limitation of computing resources. On the one hand, in Internet of Vehicles (IoVs), vehicle-to-vehicle (V2V) communications will play an important role in computation-intensive tasks processing. Since the vehicles can offload computation-intensive tasks to its one-hop neighboring vehicles, this offloading method greatly reduces the transmission delay due to the proximity between vehicles. On the other hand, there will be a growing number of vehicles possessing the stronger computation capacities in the future. Therefore, we can utilize multi-vehicle's available resources to parallelly execute tasks, which will further reduce the delay.

There are some literatures investigating the computation offloading in IoVs. In [4], by choosing neighboring vehicles as task processing helpers, a greedy algorithm was proposed to solve the optimization problem to minimize the average response time of the tasks originated from vehicles. Zhou et al. [5] proposed a parking edge computing strategy, which made use of the parked vehicles to assist edge servers in offloaded task handling. Both of the works above only selected one cooperative vehicle to complete computation-intensive tasks collaboratively, which may be not make full use of vehicles' available resources. In [6], by utilizing the gathering period of vehicles in urban environment due to stopped by traffic lights or Area of Interest (AOI), a task offloading scheme merely relying on vehicle-to-vehicle (V2V) communication was proposed by fully exploring the idle resources of gathered

This work was partially supported by Natural Science Foundation of China (Grant 61901070, 61801065, 61771082, 61871062, U20A20157), in part by the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant KJQN202000603 and Grant KJQN201900611), in part by the Natural Science Foundation of Chongqing (Grant cstc2020jcyj-zdxmX0024) and in part by University Innovation Research Group of Chongqing (Grant CXQT20017). in part by the General Project of Natural Science Foundation of Chongqing(Grant cstc2021jcyj-msxmX0892).

vehicles for task execution. Gong et al. [7] proposed three task scheduling algorithms, the min-min, the max-min and the HEFT algorithms to investigate the effectiveness of computation offloading in IoVs. However, dynamic IoVs requires to solve NP-hard combinatorial optimization problems quickly within the period time, which will be hardly achieved with traditional numerical optimization methods.

Based on the aforementioned analysis, we need to introduce intelligent strategy to manage the communication and computing resources effectively so as to cover the shortage of traditional model-based resource management methods. A basic idea of this paper is to design an offloading strategy, which decides not only where to offload the tasks, but also includes how many tasks to offload. To satisfy the requirement of low delay, we propose a multi-vehicle intelligent collaborative computing framework, which makes full use of available computing resources of adjacent vehicles. Specifically, by jointly considering the local execution, V2V offloading and multi-vehicle collaboration computation, the proposed strategy utilizes parallel computing to reduce the total system delay. Considering the V2V link quality, vehicles' mobility and available computing resources, we model the computation offloading as a decision-making problem with the goal of maximizing the cumulative reward. Then, we determine the optimal task partition ratio after the cooperative vehicles are selected. Furthermore, double deep Q-network (DDQN) is used to take the optimal dual actions. Simulation results verify that the proposed strategy can effectively allocate the vehicles' available computing resources to all tasks in the dynamic IoVs.

The rest of the paper is organized as follows. Section II presents the system model. The optimization problem is formulated in Section III. Section IV describes the proposed multi-vehicle intelligent collaborative computing strategy. We provide the numerical results in Section V. Section VI concludes the paper.

II. SYSTEM MODEL

In this section, we first present the vehicle mobile computing (VMC) system model, then give task partition model. Finally, we describe the delay model in three parts: local computing, V2V communication and cooperative vehicle computing.

A. VMC System Model

We consider a bidirectional road as illustrated in Fig. 1, there is one task vehicle which does not have enough computing resources to accomplish its tasks locally, thus requires to offload part of its tasks to neighboring vehicles that have available computing resources to provide computation service for task vehicle. The vehicles which have available computing resources to be shared are classified as cooperative vehicles. To extend the computing service range of edge computing, we select cooperative vehicles to collaboratively execute the delay-sensitive and computation-intensive tasks with task vehicle. Among them, we assume that the vehicles

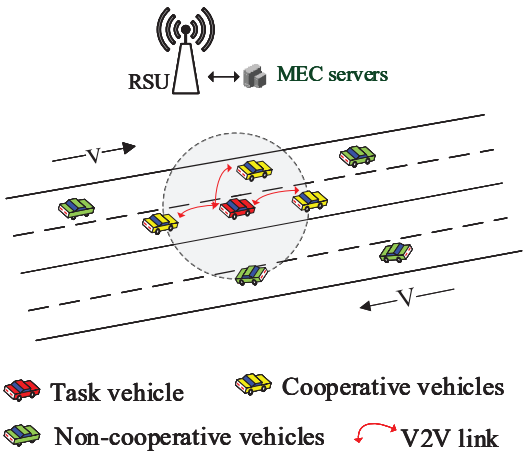

Fig. 1. Network model.

are slow moving and mostly traveling in a group. Here, we define the task vehicle as s, the set of cooperative vehicles is $\mathcal{K} = \{1, ..., k, ..., K\}$. In order to represent the V2V association, we introduce a binary indicator variable $\alpha_k \in \{0, 1\}$, $\alpha_k = 1$ indicates that the vehicle k is selected as a cooperative vehicle to accomplish task with the task vehicle, otherwise $\alpha_k = 0$.

B. Task Partition Model

We assume that vehicle s always has a delay-sensitive and computation-intensive task T_1, defined as $T_1 = \{d_1, c_1, \tau_1\}$. Here d_1 denotes the data size of T_1, c_1 denotes the total number of CPU cycles required to complete T_1, τ_1 denotes the maximum tolerable delay of T_1. In addition, within each slot, cooperative vehicle k needs to execute its own local task T_2, which arrives as a random model. We define the task T_2 as $T_2^k = \{d_2^k, c_2^k, \tau_2^k\}$, where d_2^k denotes the data size of T_2, c_2^k denotes the total number of CPU cycles required to complete T_2, τ_2^k denotes the maximum tolerable delay of T_2.

In order to minimize the total delay of the IoVs, we consider simultaneously utilizing the resources of both task vehicle and cooperative vehicles, so as to benefit from parallel computing. We assume that T_1 is of bit independence and can be partitioned into $K + 1$ subtasks, one of which is executed locally by the vehicle s itself and the other can be offloaded to the cooperative vehicles. Let $0 \leqslant \lambda_{s,0} \leqslant 1$ denotes the partition ratio of task T_1 that is executed locally, and $0 \leqslant \lambda_{s,k} \leqslant 1$ denote the partition ratio of T_1 that is offloaded to cooperative vehicle k. The sum of the partition ratios of task T_1 should be equal to 1:

$$\lambda_{s,0} + \sum_{k \in \mathcal{K}} \lambda_{s,k} = 1 \tag{1}$$

C. Delay Model

The total system delay of IoVs includes three parts: local computing delay, V2V communication delay and cooperative vehicles computing delay. We will described them in detail:

1) Local computing

We assume that task has a fixed workload, that is, the CPU cycles required to execute per bit of task T_1 are the same. We denote f^l as the CPU frequency, that is the number of CPU cycles per second of vehicle s. Thus, the local execution delay L^{loc} of task $\lambda_{s,0}T_1$ is

$$L^{loc} = \frac{\lambda_{s,0}c_1}{f^l} \tag{2}$$

Adopt the energy consumption model per CPU cycle as $e = bf^2$ [8], where b is a coefficient depending on the chip architecture, f is CPU frequency. The local execution energy consumption E^{loc} can be given as

$$E^{loc} = b(f^l)^2\lambda_{s,0}c_1 \tag{3}$$

2) V2V communications

Let the location of the vehicle s and the cooperative vehicle k are (x_s, y_s) and (x_k, y_k), respectively. The global information of IoVs is updated periodically, including vehicle location, driving direction, network status, etc. We assume that the vehicle s moves at a certain speed and in a constant direction. Then the distance between the vehicle s and the cooperative vehicle k can be expressed as

$$d_{s,k} = \sqrt{(x_s - x_k)^2 + (y_s - y_k)^2} \tag{4}$$

The uplink rate $r_{s,k}$ between vehicle s and cooperative vehicle k is

$$r_{s,k} = W\log_2(1 + \frac{p(d_{s,k}^{-\alpha})|h_0|^2}{N_0}) \tag{5}$$

where W is the uplink channel bandwidth. p is the transmission power of vehicle s. Moreover, $\alpha \in [2, 5]$ is the path-loss exponent, and h_0 is the complex Gaussian channel coefficient that follows the complex normal distribution. N_0 is the additive white Gaussian noise.

We define $L_{s,k}^{up}$ as the transmission delay. It is the time that vehicle s offloads task $\lambda_{s,k}T_1$ to cooperative vehicle k, can be expressed as

$$L_{s,k}^{up} = a_k\frac{\lambda_{s,k}d_1}{r_{s,k}} \tag{6}$$

The cooperative vehicles' choice and the partition ratio determining of task T_1 would affect the energy consumption of vehicle s. Therefore, the energy consumption of vehicle s offloading task $\lambda_{s,k}T_1$ to cooperative vehicle k is represented as

$$E_{s,k}^{coo} = a_k\frac{\lambda_{s,k}d_1}{r_{s,k}} \cdot p \tag{7}$$

3) Cooperative vehicle computing

We assume that each cooperative vehicle has the same CPU frequency as the vehicle s. The cooperative vehicle k parallelly executes the cooperative task $\lambda_{s,k}T_1$ and the local task T_2. Assume that the cooperative vehicle k allocates the computing resources f_1^l to execute cooperative task $\lambda_{s,k}T_1$, and allocates f_2^l to execute local task T_2. Here $f_1^l + f_2^l = f^l$. The cooperative vehicle computing delay $L_{s,k}^{com}$ is

$$L_{s,k}^{com} = a_k\frac{\lambda_{s,k}c_1}{f_1^l} \tag{8}$$

In order to minimize the computing delay of the task $\lambda_{s,k}T_1$, it is better to allocate more computing resources f_1^l to execute the task $\lambda_{s,k}T_1$ when the computing delay of the local task T_2 is guaranteed within its delay tolerance constraint. It can be given as

$$\frac{c_2^k}{f_2^l} \leqslant \tau_2^k \Rightarrow f_2^l \geqslant \frac{c_2^k}{\tau_2^k} \tag{9}$$

Hence, the cooperative vehicle k allocates the minimum computing resources to the local task T_2, represented as $f_2^l = \frac{c_2^k}{\tau_2^k}$. Therefore, the computing resources allocated by the cooperative vehicle k to the task $\lambda_{s,k}T_1$ can be taken to its maximum value, $f_1^l = f^l - f_2^l = f^l - \frac{c_2^k}{\tau_2^k}$. Thus, (8) can be transformed as

$$L_{s,k}^{com} = a_k\frac{\lambda_{s,k}c_1}{f^l - \frac{c_2^k}{\tau_2^k}} \tag{10}$$

Vehicle s offloads the task $\lambda_{s,k}T_1$ to the cooperative vehicle k, the total delay includes three parts: the transmission delay, the download delay of execution result, and the execution delay of task in the cooperative vehicle k. Since the size of the task execution result is much smaller than the input size of the task, the download delay of execution result can be ignored [9]. Therefore, we just consider the transmission delay and the execution delay in this paper. In IoVs, vehicle s offloads task $\lambda_{s,k}T_1, k \in \mathcal{K}$ to cooperative vehicles in parallel. Therefore, the sum of the transmission and execution delay of the task $\lambda_{s,k}T_1$ is

$$L^{coo} = \max_{k\in\mathcal{K}}\{L_{s,k}^{up} + L_{s,k}^{com}\} \tag{11}$$

III. PROBLEM FORMULATION

In this work, we aim to minimize the total system delay of IoVs. The task vehicle s computes the task $\lambda_{s,0}T_1$ locally, and offloads the remaining task $\sum_{k\in\mathcal{K}}\lambda_{s,k}T_1$ to the nearby cooperative vehicles. Since the local computing can be performed simultaneously with the multi-vehicle collaborative computing, the total delay of the system can be determined by the longer process. The total delay of the system is defined as

$$L = \max\{L^{loc}, L^{coo}\} \tag{12}$$

Based on the VMC system model, the optimization problem that minimizes the total delay of system within latency and

energy constraints can be modeled as

$$\begin{aligned} \min_{\alpha_k, \lambda_{s,k}} \ & L \\ \text{s.t. } & C1: L \leqslant \tau_1 \\ & C2: E^{loc} + \sum_{k \in \mathcal{K}} E^{coo}_{s,k} \leqslant E_s \\ & C3: \alpha_k \in \{0, 1\} \\ & C4: 0 \leqslant \lambda_{s,0} \leqslant 1 \\ & C5: \lambda_{s,0} + \sum_{k \in \mathcal{K}} \lambda_{s,k} = 1 \end{aligned} \tag{13}$$

where (13) is the goal of minimizing the total system delay. The constraint $C1$ indicates that the total delay should not exceed the maximum tolerable delay of T_1. The constraint $C2$ means that the sum of the energy consumption should not exceed the total energy of vehicle s. In $C3$, $a_k = 1$ means that the vehicle k is selected as a cooperative vehicle to accomplish task with vehicle s, otherwise $a_k = 0$. The constraint $C4$ indicates the value range of the partition ratio of T_1. The constraint $C5$ indicates that the sum of the partition ratios of T_1 is 1.

IV. MULTI-VEHICLE INTELLIGENT COLLABORATIVE COMPUTING STRATEGY

In this section, we propose a multi-vehicle intelligent collaborative computing strategy based on DRL. We will pre-select the cooperative vehicles and further determine the optimal task partition ratio calculated on local and cooperative vehicles after selecting the cooperative vehicles. The algorithm of multi-vehicle intelligent collaborative strategy is described as follows:

First, we need to reduce the input dimension of the Q-network. In a bidirectional road with large traffic, vehicle will encounter a large number of vehicles. We consider the influences of location, direction and available computing resources of neighboring vehicles, and pre-selecting the appropriate vehicles place in the cooperative vehicle set.

Then, because the optimization problem (13) is an NP-hard problem, we propose an optimization algorithm based on DDQN to solve this problem. We determine the optimal task partition ratio calculated on local and cooperative vehicles after selecting cooperative vehicles by taking dual actions through double deep Q-network. The entire process is represented in algorithm 1.

The DQN only has one network to select and evaluate actions, which generally leads to overestimate the Q-value of the action, that is, the overestimation problem [10]. The estimation error will be increased with the number of actions. If the overestimation is not uniform, we can only find the sub-optimal solution, not the optimal solution [11]. In order to solve the above problem, we introduce DDQN to solve the overestimation problem by using the main Q-network and the target Q-network to decouple the selection of action and the calculation of the target Q-value. The task vehicle acts as an agent, collecting initial observation state s_1 from

Algorithm 1 Multi-Vehicle Intelligent Collaborative Algorithm Based on DDQN

```
Generate simulation environment
Phase 1: Pre-select collaborative vehicle
for i = 1, 2, ..., I do
  if d_{s,i} ⩽ r, o_s = o_i and f_i^l > 0 then
    Put vehicle i into set K.
    Abandon vehicle i.
  end if
end for
Phase 2: Multi-VMS collaboration
for each episode do
  Update vehicle operating environment.
  for each step do
    Collect initial observation state s_1.
    Choose action a(t) from s_1 according to greedy policy,
    determine a_k(t) and λ_{s,k}.
    Calculate reward R(t+1), the state transits from s_t
    to s_{t+1} .
    Store {s_t, a(t), R(t), s_{t+1}} in the replay memory M.
    Sample mini-batch of B from M.
    Calculate the loss to update the parameters of main
    Q-network, using RMSprop.
  end for
end for
```

IoVs to input the neural network, which can obtain the value function of all actions. For each step, agent explores the state-action space by adopting soft strategy such as greedy algorithm. According to the vehicular environment dynamic, agent collects $\{s_t, a(t), R(t), s_{t+1}\}$ and stores it in the replay memory $\mathcal{M}$.

Each period, we can update the parameters of main Q-network by uniformly sample mini-batch of $\mathcal{B}$ from $\mathcal{M}$ to complete the training of the model. Firstly, the target Q-value is calculated as

$$\begin{aligned} y_t^{DDQN} &= R_{t+1} \\ &+ \eta Q_{target}(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta_t^-) \end{aligned} \tag{14}$$

Then, we can calculate the loss function according to (15). Next, we adopt RMSProp optimization algorithm and back propagation to calculate the gradient of the loss relative to the parameters of main Q-network, and change the network parameters in the opposite direction of the gradient, so as to reduce the loss. Finally, the optimal parameters in the model are found to minimize the loss value, and the predicted value of the algorithm is fitted to the target value of the input sample, which means that the training of the model is completed.

$$L(\theta) = \frac{1}{\mathcal{B}} \sum_{\mathcal{B}} (y_t^{DDQN} - Q_{main}(s_t, a_t; \theta))^2 \tag{15}$$

Key elements of the multi-vehicle intelligent collaborative strategy based on DDQN are described below in detail.

1) State

The state space can be expressed as:

$$s(t) = \{(x, y), o, T, f_1^l\} \tag{16}$$

where (x, y) includes the location of vehicle s and cooperative vehicles. o is the motion direction of vehicle s and cooperative vehicles. T represents the task T_1 originated from vehicle s and the task $T_2^k(t), k \in \mathcal{K}$ originated from cooperative vehicles. And the computing resources allocated by the cooperative vehicle k to the task $\lambda_{s,k}T_1$ is denotes as f_1^l.

2) Action

The action space can be expressed as:

$$a(t) = \{(a_1(t), \lambda_{s,1}(t)), (a_2(t), \lambda_{s,2}(t)), ..., (a_k(t), \lambda_{s,k}(t))\} \tag{17}$$

where $a_k(t) = 1$ means that the vehicle k is selected as a cooperative vehicle to accomplish task with vehicle s, otherwise $a_k(t) = 0$. $\lambda_{s,k}(t)$ is the optimal ratio of T_1 offloaded to cooperative vehicle k.

3) Reward

Reinforcement learning can flexibly design the rewards, therefore, it can be used to solve the optimization problem that is difficult to solve by traditional optimization method. When the reward is associated with the optimization goal, the system performance can be improved. The goal of this work is to minimize the total delay of system in IoVs, however, RL would maximize the reward. Therefore, the reward can be set as the inverse of the objective function. Here, the value of the denominator is greater than zero. Therefore, we set the reward for each time step as:

$$R = \begin{cases} \frac{1}{L}, & E^{loc} + \sum_{k \in \mathcal{K}} E_{s,k}^{coo} \leqslant E_s \\ -\infty, & E^{loc} + \sum_{k \in \mathcal{K}} E_{s,k}^{coo} > E_s \end{cases} \tag{18}$$

V. SIMULATION RESULTS

In this section, we evaluate the proposed multi-vehicle intelligent collaborative strategy to verify the effectiveness of the strategy. The system parameters [12] are listed in Table I. In order to analyze the impact of the proposed strategy in the IoVs, the following two benchmark strategies are used:

1) Local execution: in this case, the task T_1 is totally executed locally by the vehicle s.

2) Random offloading: in this case, for each step, the agent randomly selects the cooperative vehicles and task partition ratio.

Figure 2 shows the loss per training episode as training iteration increases to investigate the convergence property of the multi-vehicle intelligent collaborative strategy. With the training progresses, the average loss per episode continuously decreases, which demonstrates the effectiveness of the proposed training algorithm. When the training episode approximately reaches 150, the performance gradually converges. The convergency of the DDQN algorithm is one of the crucial properties to obtain a policy which maps states to the optimal actions.

Figure 3 shows the average delay comparison in different task input sizes. The CPU frequency of vehicle s is set to 140MHz. The green asterisk curve indicates the maximum

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Communication range of task vehicle r	500m
Date size of task T_1	200Kbit
Computation complexity of task T_1	1000cycle/bit
Maximum tolerable delay of task T_1	6ms/Kbit
CPU frequency of vehicle s	140MHz
Coefficient depending on the chip architecture b	10^{-26}
Bandwidth	100MHz
Vehicle transmission power	23dBm
Noise power σ^2	-114dBm
Path loss exponent	3

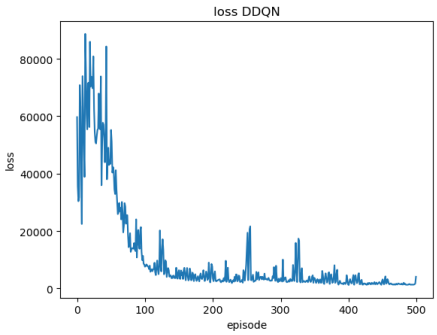

Fig. 2. Loss for each training episode.

tolerable delay of the task T_1 , which is proportional to the data size of T_1. Simulation result indicates that the average delay increases as the data size increases. this is because the computing delay and transmission delay of the task are proportional to the data size of task.When the date size of T_1 is 200Kbit, the system delay of the proposed strategy is 0.44s, and the delays of local execution and random offloading strategies are 1.44s and 1.07s respectively. The proposed strategy has lower delay, because the larger date size of task when the CPU frequency of vehicle s remains unchanged, the task T_1 is divided into multiple parts for parallel executing, and better average delay performance can be obtained. It can

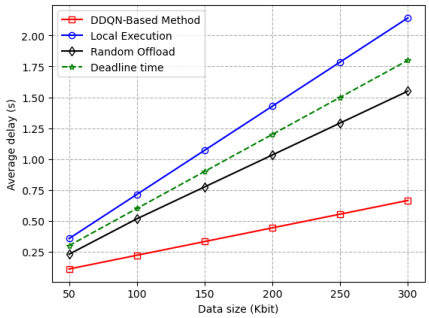

Fig. 3. Average delay versus date size.

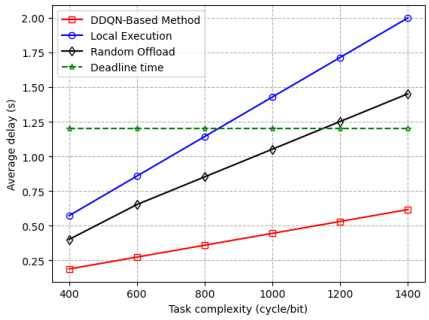

Fig. 4. Average delay versus task complexity.

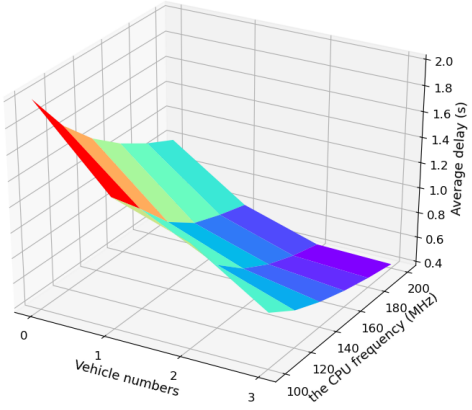

Fig. 5. Average delay versus the number of vehicles and CPU frequency.

also see that the average delay of the local execution is always greater than the tolerable delay of task T_1 . It indicates that the computing capacity of vehicle s is not enough to execute all task locally.

Figure 4 shows the influence of the task complexity on the delay performance. The delay of three strategies increases with the change of task complexity. As the complexity of the task increases, the task will be very difficult to calculate, which means that more execution delays are consumed.The change of task complexity has no influence on the ratio of the number of CPU cycles required to complete the task T_1 to the task complexity, which means that the data size of task T_1 has not changed. Furthermore, the average delay of the proposed strategy is 0.44s, which is improved by 69.4% and 58.9% compared with the local execution and random offloading, respectively.

Figure 5 shows the influence of both the number of cooperative vehicles and the CPU frequency of vehicle s on the average delay. When the number of vehicles is fixed, the average delay decreases with the increase of CPU frequency. When the CPU frequency of vehicle s is fixed, with more and more cooperative vehicles, we can make full use of the advantages of parallel computing to reduce total delay. Compared with local execution, that is, when the number of cooperative vehicles is 0, the proposed multi-vehicle intelligent collaborative strategy can effectively reduce the system average delay.

VI. CONCLUSION

In this paper, we have investigated the problem of efficiency computation offloading for delay-sensitive and computation-intensive applications in IoVs. To minimize the total system delay, we make full use of available computing resources of adjacent vehicles, and utilize parallel computing by jointly considering the local execution, V2V offloading, and multi-vehicle collaboration. Considering the dynamics of vehicles' location and service request, we propose a multi-vehicle intelligent collaborative computing strategy based on DDQN to determine the optimal task partition ratio after the cooperative vehicles are selected by taking the optimal dual actions. Simulation results have demonstrated that our proposed strategy has better performance compared to the other benchmark strategies. In our future work, we will consider the trade-off between the energy consumption and application completed time of vehicles.Further, we will investigate the multi-platform collaborative computing strategy by considering more practical environment for IoVs.

REFERENCES

[1] H. Zhao, Q. Zhu, Y. Chen and Y. Zhu, "A Research of Task-Offloading Algorithm for Distributed Vehicles," 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 2020, pp. 1-5.

[2] A. Yousafzai, I. Yaqoob, et al., ¡°Process migration-based computational offloading framework for iot-supported mobile edge/cloud computing,¡± IEEE Internet of Things Journal, vol. 7, no. 5, 2020, pp. 4171¨C4182.

[3] W. Saad, M. Bennis, et al., ¡°A vision of 6g wireless systems: Applications, trends, technologies, and open research problems, IEEE Network, vol. 34, no. 3, 2020, pp. 134¨C142.

[4] Y. Wu, J. Wu, L. Chen, G. Zhou and J. Yan, "Fog Computing Model and Efficient Algorithms for Directional Vehicle Mobility in Vehicular Network," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.2971343.

[5] C. Ma, J. Zhu, M. Liu, H. Zhao, N. Liu and X. Zou, "Parking Edge Computing: Parked Vehicle Assisted Task Offloading for Urban VANETs," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2021.3056396.

[6] C. Chen et al., "Delay-Optimized V2V-Based Computation Offloading in Urban Vehicular Edge Computing and Networks," in IEEE Access, vol. 8, pp. 18863-18873, 2020, doi: 10.1109/ACCESS.2020.2968465.

[7] M. Gong and S. Ahn, "Computation Offloading- Based Task Scheduling in the Vehicular Communication Environment for Computation-Intensive Vehicular Tasks," 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 2020, pp. 534-537, doi: 10.1109/ICAIIC48513.2020.9064975.

[8] W. Zhang, Y. Wen, et al. ¡°Energy optimal mobile cloud computing under stochastic wireless channel,¡± IEEE Transactions on Wireless Communications, vol. 12, no. 9, pp. 4569¨C4581, Sep. 2013.

[9] J. Zhang et al., "Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile Edge Computing Networks," in IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2633-2645, Aug. 2018, doi: 10.1109/JIOT.2017.2786343.

[10] Liang L, Ye H, Li G Y. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(10): 2282-2292.

[11] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2016, 30(1).

[12] J. Wang, T. Lv, P. Huang and P. T. Mathiopoulos, "Mobility-aware partial computation offloading in vehicular networks: A deep reinforcement learning based scheme," in China Communications, vol. 17, no. 10, pp. 31-49, Oct. 2020, doi: 10.23919/JCC.2020.10.003.