GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685717

Resource Allocation via Edge Cooperation in Digital Twin Assisted Internet of Vehicle

Tong Liu*†, Lun Tang*, Weili Wang*, Xiaoqiang He*,Qianbin Chen*, *Senior Member, IEEE*
*School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing, China
†School of Big Data and Internet of Things,
Chongqing Vocational Institute of Engineering, Chongqing, China
Email:liuyyshiwo@163.com, tangl@cqupt.edu.cn, 1961797154@qq.com,
hexiaoq_casey@foxmail.com and chenqb@cqupt.edu.cn

***Abstract*—In this paper, we propose a Digital Twin (DT) Supported Resource Allocation Scheme (DTS-RAS), which empowers the intelligent edge cooperation in the Internet of Vehicles (IoV) environment possible. We focus on the latency minimization under the DT-IoV framework. Specifically, we formulate the mathematical expression for the response time of vehicle offloading tasks to cooperative edge nodes according to modeling the edge server as a M/M/1/N queen. Then, we construct the optimization model aiming at reducing the response time. In view of the complexity, we apply a Double Deep Q-learning Network (DDQN) to training the edge server to get an optimal allocation action by modeling the cooperation process as an MDP. Simulation results demonstrate that our proposed scheme outperforms the existing schemes in terms of execution latency.**

***Index Terms*—Edge cooperation; Digital twin; Artificial Intelligent; Resource allocation**

I. Introduction

THE applications in the Internet of Vehicles (IoV) are characterized by latency-sensitive which are difficult to implement under the current network environment [1] [2]. Deploying Edge Servers (ESs) at the edge of the network effectively shortens the distance from the vehicle to the data processing center, thereby reducing the network response time [3] [4].

In addition, edge collaboration has been proven to be an effective way to improve system performance [5]. In the edge collaboration IoV environment, the load balance of the network can be achieved through reasonable allocation of communication, computing, and cache (3C) resources between edge nodes, thereby improving the utilization of network resources. However, 3C resources allocation in IoV faces a few critical issues such as the stochastic nature of wireless channels, the high mobility of vehicles and the limited radio and computing resources shared by multiple vehicles. These problems together increased the difficulty of resource allocation in the IoV. One promising means that can solve these matters synchronously is Artificial Intelligent (AI) [6] [7]. Lots of outstanding works proved that AI is a valid tool that can improve resource utilization and realize optimal resource allocation in IoV scenarios [8] [9]. However, most of the existing researches ignore the fact that the edge collaboration and resource allocation supported by AI require strong computing capabilities of the ESs. The inadequate computing capability of the ESs results in insufficient training of the neural network, which further affects the accuracy of its prediction.

A feasible solution to break the bottleneck is constructing a Digital Twin (DT) empowered IoV architecture by introducing DT technology into IoV. By sending traffic data of vehicle and operating data of communication equipment such as Road Side Units (RSUs) and Macro Base Station (MBS) to the DT data center, a DT of each physical entity and communication environment is established. DT can be deployed in data collection centers to configure and arrange the resources of the IoV from a global perspective. At present, the research on DT-IoV is still in the early stage. In [10], DT is used to optimize the Quality of Service (QoS) of the service. An effective and adaptable service offloading scheme is introduced to optimize the QoS. Authors in references [11] noticed the existence of deviation between the real world and the digital representation. Therefore, in literature [11], they conduct mathematical derivation to analyze the differences in system performance caused by the bias of computing resource.

In this paper, we employ DT to assist edge collaboration in IoV to achieve the purpose of minimizing task processing delay through the reasonable allocation of 3C resources. In addition, we conduct a detailed analysis of the impact of data deviation on system performance. As far as we know, this paper is the early effort on edge collaboration in DT-IoV.

II. System Model

As shown in Figure 1, the system includes a MBS, lots of RSUs, ESs and a single vehicle running on the road. The vehicle is equipped with sensors to monitor the status data such as the speed of the vehicle, the distance between the vehicle and the roadside, etc.. The RSUs installed on the roadside provide communication access to the vehicle. ESs provide vehicle with high-reliability and low-latency services are installed at each RSU. In this system, all RSUs are connected with ESs through optical fibers through which the RSUs are also connected. In addition, the RSUs also send the current data of the ESs and themselves, such as the busy/idle status of the ES, remaining resources to the MBS through the real-time channel. The MBS is the data storage center of the

DT. Except providing traditional base station services, it also stores the information of all devices within its coverage area. The vehicle also sends its own current operational data to the MBS to build the DT model.

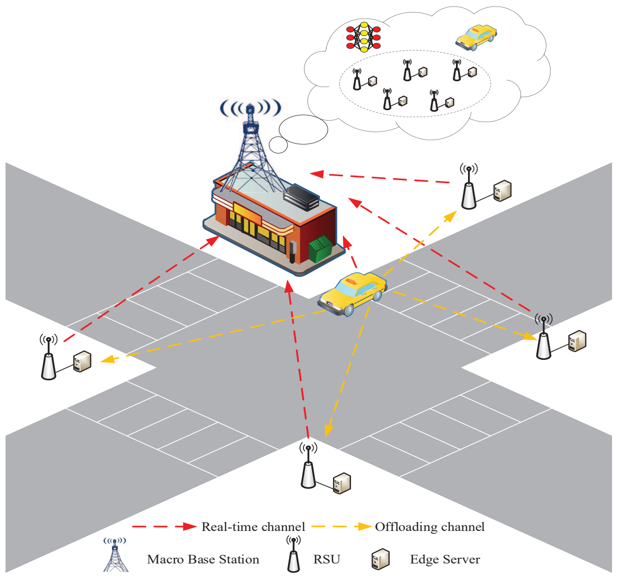

Fig. 1: System model.

In addition, the connection and communication status between each device also own its digital representation in the DT space. It is worth noting that due to the fading characteristics of the wireless channel, there must be deviations between the DT and the real world. In this paper, these deviations are set to be controllable to ensure the fidelity of the DT.

The task of vehicle in the physical world can be offloaded to the RSUs searching for the assistance processing of ESs through offloading channel. The ES which receive the task may be currently in a busy state, so the task can be further distributed to other ESs for cooperative execution. In order to minimize the response time, ES needs to comprehensively consider the current available communication, computing and storage resources of other ESs to determine the amount of tasks to be offloaded. The massive historical data stored in DT can provide raw data for neural network training to assist the ES to determine the proportion of the distribution task.

III. Performance Analysis

The number of ESs is m, forming an edge server set $E = \{e_1, e_2, \ldots, e_m\}$. The task of the vehicle is expressed as $W = \{D_n, C_n, T_{th}\}$. D_n is the data volume of the task. C_n means the CPU cycles required to perform the task and T_{th} denotes the maximum tolerable delay for this task. It is assumed the task W is divided into n parts. The proportion coefficient of each subtask to the total task is α_j and can be denoted as $M_j = \left\{D_n^j, C_n^j, T_{th}^j\right\} j = 1, 2, \ldots, n$. D_n^j is the data amount of subtask M_j, satisfying $D_n = \sum_{j=1}^{n} D_n^j$.

All information of RSUs, ESs, vehicle and the communication environment connecting them are stored in the DT server in the MBS. Moreover, the data of ESs is cached in matrix $\tilde{Z}$ with the size of $m \times n$. The digital twin data of ESs at time t can be expressed as

$$\tilde{Z}(t) = \begin{bmatrix} \tilde{z}_{11}(t) & \tilde{z}_{12}(t) & \cdots & \tilde{z}_{1n}(t) \\ \tilde{z}_{21}(t) & \tilde{z}_{22}(t) & \cdots & \tilde{z}_{2n}(t) \\ \vdots & \vdots & \cdots & \vdots \\ \tilde{z}_{m1}(t) & \tilde{z}_{m1}(t) & \cdots & \tilde{z}_{mn}(t) \end{bmatrix}_{m \times n} \tag{1}$$

m denotes the number of ESs, n represents the attributes of each ES. To facilitate analysis, defining the first, the second and the third column of the matrix $\tilde{Z}$ to represent the remaining communication, computing and storage resources of each ES at time t. So we have

$$\begin{cases} \tilde{Z}_1(t) = \begin{bmatrix} \tilde{z}_{11}(t) & \tilde{z}_{21}(t) & \cdots & \tilde{z}_{m1}(t) \end{bmatrix}^{\mathrm{T}} \\ \tilde{Z}_2(t) = \begin{bmatrix} \tilde{z}_{12}(t) & \tilde{z}_{22}(t) & \cdots & \tilde{z}_{m2}(t) \end{bmatrix}^{\mathrm{T}} \\ \tilde{Z}_3(t) = \begin{bmatrix} \tilde{z}_{13}(t) & \tilde{z}_{23}(t) & \cdots & \tilde{z}_{m3}(t) \end{bmatrix}^{\mathrm{T}} \end{cases} \tag{2}$$

As for other information, such as the current remaining energy of the ESs, are stored in other columns. The loss of data during the transmitting process causes deviations between the data stored in the DT and the real world. $\Delta Z(t)$ is defined to represent these data gaps, expressed as

$$\Delta Z(t) = \begin{bmatrix} \Delta z_{11}(t) & \Delta z_{12}(t) & \cdots & \Delta z_{1n}(t) \\ \Delta z_{21}(t) & \Delta z_{22}(t) & \cdots & \Delta z_{2n}(t) \\ \vdots & \vdots & \cdots & \vdots \\ \Delta z_{m1}(t) & \Delta z_{m2}(t) & \cdots & \Delta z_{mn}(t) \end{bmatrix}_{m \times n} \tag{3}$$

Also, the first, second and third columns of $\Delta Z(t)$ are respectively defined as the communication, computing and storage resource gaps of the ESs in keeping with the above definition, which can be denoted as

$$\begin{cases} \Delta Z_1(t) = \begin{bmatrix} \Delta z_{11}(t) & \Delta z_{21}(t) & \cdots & \Delta z_{m1}(t) \end{bmatrix}^{\mathrm{T}} \\ \Delta Z_2(t) = \begin{bmatrix} \Delta z_{12}(t) & \Delta z_{22}(t) & \cdots & \Delta z_{m2}(t) \end{bmatrix}^{\mathrm{T}} \\ \Delta Z_3(t) = \begin{bmatrix} \Delta z_{13}(t) & \Delta z_{23}(t) & \cdots & \Delta z_{m3}(t) \end{bmatrix}^{\mathrm{T}} \end{cases} \tag{4}$$

Other data bias is stored in the columns corresponding to the attributes in the matrix $\Delta Z(t)$.

When the task exceeds its own computing ability, the vehicle should offload it to a ES. From the perspective of reducing the transmitting delay, the DT allocates the ES with the most abundant communication resource to vehicle for access. So, the selected access ESk can be expressed as

$$k = \arg\max[z_{11}, z_{21} \ldots z_{m1}] \tag{5}$$

After receiving the task, ESk divides the task into subtasks and sends them to other ES j with the assistance of DT by combining the current state of ES j, such as the current remaining resources, whether the software necessary to solve the task is cached, etc.. If ESk itself has sufficient resources, it can also feasible to process subtask locally. After queuing, subtasks can be solved by ESj installed in RSUj. The results then will be sent to the vehicle. Since the amount of feedback data is quite little, we set it as 0 in our research [9].

Since the storage space of the task is further reduced after the task is divided, it is assumed that the ESj has sufficient cache space for storing subtasks M_j, satisfying $z_{j3}(t) \geq D_n^j (\forall j \in M)$, where $z_{j3}(t) = \tilde{z}_{j3}(t) + \Delta z_{j3}(t)$ is the actual remaining storage resources of ESj. C_n^j is the CPU cycle required to execute the task M_j, satisfying $C_n = \sum_{j=1}^{n} C_n^j$. $T_{th} = \max(T_{th}^1, T_{th}^2, \ldots, T_{th}^j)$ is the tolerable delay threshold.

According to the task processing flow, the total time required to solve the task can be expressed as

$$T_t(t) = T_{trans}^{v\to k}(t) + \max(T_{trans}^{k\to j}(t) + T_{que}^j(t) + T_{soft}^j(t) + T_{exe}^j(t)) \quad (6)$$

$T_{trans}^{v\to k}(t)$ is the transmitting time required for vehicle sending task to ESk. $T_{trans}^{k\to j}(t)$ means the distribution time for ESk send subtasks to ESj. Queuing delay is denoted as $T_{que}^j(t)$. $T_{soft}^j(t)$ represents the time consumed in downloading the software. $T_{exe}^j(t)$ is used to indicate the executing time.

The time required to send the task from the vehicle to the ESk is expressed as

$$T_{trans}^{v\to k}(t) = D_n / s_{v\to k} \quad (7)$$

$s_{v\to k}$ denotes the channel rate between vehicle to ESk and can be calculated by

$$s_{v\to k} = z_{k1}(t) \log_2(1 + \frac{p_v h_v}{N_v + I_v}) \quad (8)$$

$z_{k1}(t)$ is the bandwidth ESk allocated to vehicle. p_v stands for the transmit power of vehicle. h_v denotes the channel gain. N_v and I_v represent the noise and interference respectively.

Due to the existence of deviation, the real communication resource allocated to the vehicle should be expressed as

$$z_{k1}(t) = \tilde{z}_{k1}(t) + \Delta \tilde{z}_{k1}(t) (k \in M) \quad (9)$$

where $\tilde{z}_{k1}(t)$ and $\Delta \tilde{z}_{k1}(t)$ is the estimated and deviation value of communication resource allocated to vehicle. Therefore, the real time consumed for sending task is equal to the sum of the estimated and the deviation transmission time, which can be express as

$$T_{trans}^{v\to k}(t) = \tilde{T}_{trans}^{v\to k}(t) + \Delta T_{trans}^{v\to k}(t) \quad (10)$$

$\tilde{T}_{trans}^{v\to k}(t)$ is the estimated transmission time calculated based on the communication resources stored in DT, $\Delta T_{trans}^{v\to k}(t)$ is the bias and represented by

$$\Delta T_{trans}^{v\to k}(t) = \frac{-\Delta z_{k1}(t))}{\tilde{z}_{k1}(t) \times (\tilde{z}_{k1}(t) + \Delta z_{k1}(t))} \times \frac{D_n}{\log_2(1 + \frac{p_v h_v}{N_v + I_v})} \quad (11)$$

Furthermore, the time required for sending the task in the real communication environment should be expressed as

$$T_{trans}^{v\to k}(t) = (\frac{1}{\tilde{z}_{k1}(t) + \Delta z_{k1}(t)}) \times \frac{D_n}{\log_2(1 + \frac{p_{v,k} h_{v,k}}{N_{v,k} + I_{v,k}})} \quad (12)$$

Assuming the ESs participating in the collaboration constitute a set $J = \{1, 2, \ldots, j\}$. Then, the distribution time consumed by the ESk to send the task to the collaborative ESj can be computed by

$$T_{trans}^{k\to j}(t) = \alpha_j D_n d_{k\to j} / \upsilon_f(t) \quad (13)$$

$\upsilon_f(t)$ is the transmission rate in the optical fiber. $d_{k\to j}$ is the distance from ESk to ESj, which can be calculated by

$$d_{k\to j} = \sqrt{(lon_k(t) - lon_j(t))^2 + (lat_k(t) - lat_j(t))^2} \quad (14)$$

$lon_k(t)$ and $lat_k(t)$ represent the longitude and latitude of ESk. $lon_j(t)$ and $lat_j(t)$ respectively denote the longitude and latitude of the ESj.

It is assumed that each ES can provide services to a single user at the current moment. At the same time, the computing capacity provided by the ES during the service period is limited. Therefore, the ESj processing task model can be modeled as the $M/M/1/N$ FCFS queuing model according to the queuing theory. N is the upper limitation of the computing ability of the ESj.

Task arrival follows a Poisson distribution with arrival rate λ_j. μ_j is adopted to represent the service rate of ESj. $\rho_j = \lambda_j / \mu_j$ is used to measure how busy the ESj is. The queuing time in the ESj is equal to the sum of the estimated and the bias value due to the deviation of computing resource, which can be expressed as

$$T_{que}^j(t) = \tilde{T}_{que}^j(t) + \Delta T_{que}^j(t) \quad (15)$$

According to the relevant knowledge of queuing theory, the expression of estimated queuing time can be calculated by

$$\tilde{T}_{que}^j(t) = \frac{\mu_j^{\tilde{z}_{j2}(t)}}{\lambda_j (\mu_j - \lambda_j)^{\tilde{z}_{j2}(t)}} \times \frac{\rho_j (1 - \rho_j^{\tilde{z}_{j2}(t)} (\tilde{z}_{j2}(t) + 1 - \rho_j \tilde{z}_{j2}(t)))}{(1 - \rho_j^{\tilde{z}_{j2}(t)+1})(1 - \rho_j)} \quad (16)$$

Furthermore, the queuing bias time can be denoted as

$$\Delta T_{que}^j(t) = \frac{\rho_j}{\lambda_j (1 - \rho_j)} \times (\frac{\mu_j}{\mu_j - \lambda_j})^{\Delta z_{j2}(t)} \times \left\{ \begin{array}{l} \frac{1 - \rho_j^{\tilde{z}_{j2}(t) + \Delta z_{j2}(t)} [\tilde{z}_{j2}(t) + \Delta z_{j2}(t) + 1 - \rho_j (\tilde{z}_{j2}(t) + \Delta z_{j2}(t))]}{1 - \rho_j^{\tilde{z}_{j2}(t) + \Delta z_{j2}(t) + 1}} \\ \times (\frac{\mu_j}{\mu_j - \lambda_j})^{\tilde{z}_{j2}(t)} - \frac{1 - \rho_j^{\tilde{z}_{j2}(t)} (\tilde{z}_{j2}(t) + 1 - \rho_j \tilde{z}_{j2}(t))}{1 - \rho_j^{\tilde{z}_{j2}(t)+1}} \end{array} \right\} \quad (17)$$

By taking mathematical changes to the above two formulas and substitute them into formula (15), we can get the realistic queuing time express as

$$T_{que}^{j}(t)=\frac{\rho_j}{\lambda_j(1-\rho_j)}\times\left\{\begin{array}{l}\frac{1-\rho_j^{\tilde{z}_{j2}(t)}(\tilde{z}_{j2}(t)+1-\rho_j\tilde{z}_{j2}(t))}{(1-\rho_j^{\tilde{z}_{j2}(t)+1})}\times\left(\frac{\mu_j}{\mu_j-\lambda_j}\right)^{\tilde{z}_{j2}(t)}\\+\frac{1-\rho_j^{\tilde{z}_{j2}(t)+\Delta z_{j2}(t)}[\tilde{z}_{j2}(t)+\Delta z_{j2}(t)+1-\rho_j(\tilde{z}_{j2}(t)+\Delta z_{j2}(t))]}{1-\rho_j^{\tilde{z}_{j2}(t)+\Delta z_{j2}(t)+1}}\\\times\left(\frac{\mu_j}{\mu_j-\lambda_j}\right)^{\tilde{z}_{j2}(t)+\Delta z_{j2}(t)}\\-\frac{1-\rho_j^{\tilde{z}_{j2}(t)}(\tilde{z}_{j2}(t)+1-\rho_j\tilde{z}_{j2}(t))}{1-\rho_j^{\tilde{z}_{j2}(t)+1}}\times\left(\frac{\mu_j}{\mu_j-\lambda_j}\right)^{\tilde{z}_{j2}(t)}\end{array}\right\} \tag{18}$$

The task can only be solved successfully with the support of the software. However, it is not necessary to cache the software at every ES for the purpose of saving storage resources. ESj can download the software from other ESs if this software is not cached locally. A binary number x_i is defined to indicate whether the ESj caches software or not.

$$x_j=\begin{cases}0 & \text{Software cached in } esj\\ 1 & \text{Software do not cached in } esj\end{cases} \tag{19}$$

In order to ensure that the task can be solved successfully, it is assumed that there is at least a copy of the software in all ESs, satisfying

$$\sum_{j=1}^{m} x_j \geq 1 \forall j \in M \tag{20}$$

Meantime, the software is cached in ES according to its popularity which follows the Zipf distribution and can be expressed as

$$\zeta_i = i^{-\tau} \Big/ \sum_{j=1}^{L} j^{-\tau} \tag{21}$$

L is the number of software. τ denotes the skewness coefficient of Zipf distribution. The larger the value of τ means the more frequently the software with high popularity is requested.

Furthermore, suppose the data size of the software is D_{soft}. Then, the time consumed in downloading software can be obtained by

$$T_{soft}^{j}(t) = (1-x_j)D_{soft}d_{c\to j}/\zeta v_f(t) \tag{22}$$

where $d_{c\to j}$ is the distance between the ESc which cached the software and the ESj, and can be computed by

$$d_{c\to j} = \sqrt{(lon_c(t)-lon_j(t))^2+(lat_c(t)-lat_j(t))^2} \tag{23}$$

$lon_c(t)$ and $lat_c(t)$ represent the longitude and latitude of the ESc. If the software is already cached in the ESj, the ESj does not need to download it, obviously $d_{j\to j}=0$.

The time consumed in execution depends on the computing capacity of the ESj and the CPU cycles required by the subtasks together. The true value of the computing ability of ESj can be expressed as

$$z_{j2}(t) = \tilde{z}_{j2}(t) + \Delta z_{j2}(t)(j \in M) \tag{24}$$

Then, we can obtain the real executing time by

$$T_{exe}^{j}(t) = \frac{\alpha_j C_n}{\tilde{z}_{j2}(t)+\Delta z_{j2}(t)} \tag{25}$$

IV. Problem formulation and algorithms

A. Problem formulation

In this paper, we focus on optimizing the allocation of 3C resources through the cooperation between ESs under the support of DT in IoV environment with the purpose of minimizing the response delay of the task, so the optimization model can be modeled as

$$\begin{aligned}&\min_{\alpha_i} T_t(t)\\ s.t.\ & C1:\ T_t(t)\leq T_{th}\\ & C2:\max(z_{i3}(t)+\Delta z_{i3}(t))\geq D_n \forall i\in M\\ & C3:\ \sum_{i=1}^{m} z_{i2}(t)+\Delta z_{i2}(t)\geq F_n\ \forall i\in M\\ & C4:\sum_{i=1}^{m} x_i \geq 1 \forall i\in M\\ & C5:\ \sum_{i=1}^{m}\alpha_i = 1\forall i\in M\end{aligned} \tag{26}$$

$C1$ limits the total time spent in solving the task cannot exceed the maximum tolerable delay of the task. $C2$ is set to ensure that the task can be successfully offloaded to at least one ES. $C3$ ensures that the computing resource allocated to the task is sufficient to handle the task. $C4$ is proposed to make sure that at least one ES caches the software. $C5$ is the requirement of distribution ratio to ensure that tasks can be executed completely. The calculation amount of the this model is extremely huge, and the variables are tightly coupled with each other. Therefore, we solve it with the help of the Double Deep Q-learning Network (DDQN).

B. Algorithms

The task distribution process of ESj satisfies the subject, environment, behavior, state and reward components of the Markov Decision Process (MDP), so the task distribution process can be modeled as an MDP.

$Q(s,a)$ is supposed to represent the Q function corresponding to state s and action a. The value of $Q(s,a)$ can be approximated by the parameters of the neural network, which is indicated by $Q(s,a)\approx Q(s,a;\omega)$. $Q(s,a;\omega)$ indicates the estimated value of $Q(s,a)$. ω is the set of parameters of the neural network.Defining the output of the target network is Q_t, which is expressed as $Q_t = R(s,a)+\gamma\max Q(s',a';\omega\ )$. s' stands for the next status. $\overline{\omega}$ denotes the parameter set of the target network. The loss function $L(\omega)$ can be expressed as $L(\omega)=E\left[(Q_t-Q(s,a;\omega))^2\right]$.

The stochastic gradient descent method is employed to refresh the parameters of the neural network. The gradient of the loss function can be described as

$$\frac{\partial L(\omega)}{\partial\omega} = \frac{\partial E\left[(Q_t-Q(s,a;\omega))^2\right]}{\partial\omega} \tag{27}$$

Algorithm 1 DDQN training algorithm

```
1.  Initialize $P$
2.  Randomly initialize $\omega$
3.  Set $\overline{\omega} = \omega$
4.    For $\Delta = 1, N$ do
5.      Set $j = 1$, randomly select initial input $q_1$
6.      Repeat
7.      Take action $a_j$ according to $\varepsilon - greedy$ mechanism
8.      Perform action $a_j$ and get reward $R(s_j, a_j)$
9.      Observe the next state $s_{j+1}$
10.     Update the input to $q_{j+1}$ based on the observation
11.     Save $p_j = (q_j, a_j, R(s_j, a_j), q_{j+1})$ in the experience pool
12.     Randomly select a small sample $\delta$ from the experience pool
13.     Clear set $\Omega$
14.         For $p_{\bar{j}} \in \delta$
15.           Main network output $Q(s_{\bar{j}}, a_{\bar{j}}; \omega)$
16.           Target network output $Q_t = \max Q(s_{\bar{j}+1}, a'; \overline{\omega})$
17.           Calculate $(Q_t - Q(s_{\bar{j}}, a_{\bar{j}}; \omega))^2$
18.           Store the result in set $\Omega$
19.         End for
20.     Calculate the loss function
21.     Obtain the gradient
22.     Update the main network parameters
23.     Every time interval $Z$, update the $\overline{\omega} = \omega$
24.     otherwise $\overline{\omega}$ remains unchanged
25.     $j = j + 1$
26.     Until Q matrix convergence
27.   End For
28. Save the latest $\omega$
```

The input of the main network is defined as $q_j = (s_{j-\vartheta}, ..., s_{j-1}, s_j)$, which includes the present and historical states of the system. The system state at the current time j is expressed as s_j. In addition, ϑ is a positive integer. The value range of ε is (0,1). In addition, the algorithm may miss the global optimal solution, so $\varepsilon-greedy$ strategy is employed to match the action a_j to state s_j. $\Pr(a_j)$ represents the probability of taking action a_j, which can be presented as

$$\Pr(a_j) = \begin{cases} \varepsilon & a_j = \arg\max Q(s_j, a; \omega) \\ 1-\varepsilon & else \end{cases} \tag{28}$$

After action a_j is applied, the system achieves an immediate return $R(s_j, a_j)$ and observes the following state s_{j+1} which is used to update the input of the main network to the next state q_{j+1}, and p_j is adopted to cache the data of the present state j, that is $p_j = (q_j, a_j, R(s_j, a_j), q_{j+1})$. Data p_j is stored in the experience pool, from which the system can randomly choose a single data which is expressed as $p_{\bar{j}} = (q_{\bar{j}}, a_{\bar{j}}, R(s_{\bar{j}}, a_{\bar{j}}), q_{\bar{j}+1})$.

The main network can output the estimated value $Q(s_{\bar{j}}, a_{\bar{j}}; \omega)$ of $Q(s_{\bar{j}}, a_{\bar{j}})$. The output of the target network can be denote as $Q_t = \max Q(s_{\bar{j}+1}, a'; \overline{\omega})$.

Both the main and the target network are composed of neural networks. The output layer can output the estimated Q value corresponding to each action under the current state. The input $q_{j,i}$ corresponds to a weight ω_i, and the weighted sum net of the input data is described as

$$net = \left(\sum_{i=1}^{\overline{n}} \omega_i \cdot q_{j,i} \right) - b \tag{29}$$

Algorithm 2 Optimized task distribution algorithm based on DDQN

```
1.  Initialize the set as none, $\Phi = \emptyset$
2.  Repeat
3.      Observe status information $s(t)$
4.      Store $\Phi = \emptyset$ into set $\Phi$
5.      Check whether the task needs to be offloaded
6.          If no, back to step 3
7.          If yes, go to step 8
8.      Construct the input of the main network $q_t$
9.      Input $q_t$ into the main network and calculate the result
10.     Select $\pi^*_{s(t)}$ from the output of the main network
11.     Adjust the task allocation ratio according to $\pi^*_{s(t)}$
12. Until task offloading completed
```

$\overline{n}$ is the number of inputs of the neuron. b represents the bias value. net indicates the input of the activation function $F(*)$.

The stochastic gradient descent method is used to update the network parameters, which can be expressed as

$$\omega_i^n = \omega_i - \frac{\partial L(\omega)}{\partial \omega_i} \tag{30}$$

After trained for several times in the main network of DDQN, the Q value function can be approximated by regularly updating the weight ω. As algorithm 1 shows, the trained parameter ω of main network will be obtained after the Algorithm 1 convergence. Then, Algorithm 2 describes the task distribution scheme based on Algorithm 1.

V. Performance Evaluation

The positions of the ESs are randomly distributed within a square range of 1000m. The transmit power of the vehicle is set as 0.2W. Noise and interference are set to 1. The CPU clock speed of vehicle is fixed as 2×10^{10} Hz. In addition, the value range of the CPU clock speed of ESs is $[1, 9] \times 10^{12}$ Hz. The neural network contains 4 layers, The number of neurons in the hidden and fully connected layer is 20. The value of ε is set as 0.45. The capacity of the experience pool is 8000.

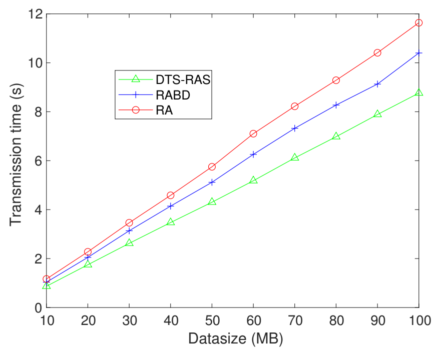

Fig. 2: Comparison on transmitting time.

Then, we analyze the transmission delay under different communication resource allocation schemes. The benchmark are Random Allocation (RA) and Resource Allocation Based on Distance (RABD). From Figure 2 we can see RABD is better than RA in transmitting latency performance. For example, when the transmission task volume is 50MB, the transmission time required by RABD is 4.9 s, which saves 0.7s compared with the 5.6s of RA. However, when transmitting tasks with the same amount of data, DTS-RAS can save 0.8s of time on the basis of RABD.

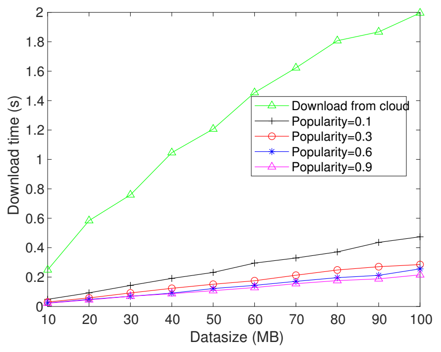

Fig. 3: Comparison on download time.

We compare the time spend in caching software under different popularity in Figure 3. It can be seen that without ES cooperation, downloading software in the cloud takes the longest time. For example, when the task volume is 60MB, the time for download is 1.45s. In addition, as the software becomes more and more popular, DTS-RAS can effectively save more cache time. For example, when the task volume is 60MB and the software popularity is 0.1, the average time required to download the software is 0.27s. As the amount of ESs cached software increases, for instance, when the software popularity is 0.9, the average time spend in downloading the software guided with DTS-RAS is only 0.12s.

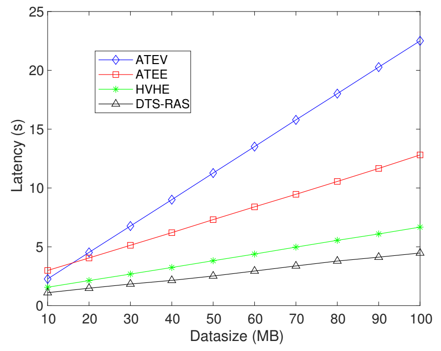

Fig. 4: Comparison on total time.

Finally, we compare the DTS-RAS with All Task Executed in Vehicle (ATEV), All Task Executed in ES (ATEE) and Half in Vehicle Half in ES (HVHE). It can be clearly seen from Figure 4 that the solution proposed in this article is significantly better than the other three schemes. For example, when the task data volume is 50MB, the response time required by DTS-RAS only takes 2.5s, while ATEV, ATEE and HVHE consume 11.2s, 7.3s and 3.8s respectively. Compared with them, DTS-RAS can save 8.7s, 4.8s and 1.3s respectively.

VI. Conclusions

In this paper, we study the problem of achieving edge collaboration through 3C resource allocation with the assistance of DT in the IoV environment to minimize task processing delay. We first analyze the response time it takes for vehicles to offload tasks to the edge nodes for collaborative processing. Secondly, we focus on the impact of the deviation between the stored data of the 3C resources in DT and the real data on the performance of the collaborative system, so as to seek the optimal resource allocation scheme. Furthermore, we establish a mathematical optimization model aimed at minimizing the system delay, which is then solved by DDQN. Future work will focus on the model building in DT-IoV.

Acknowledgment

This work is partially supported by National Natural Science Foundation of China (Grant No. 62071078), the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJZD-M201800601) , the Major theme special projects of Chongqing (Grant No. cstc2019jscx-zdztzxX0006) and the Key scientific research project of Chongqing Vocational Institute of Engineering (Grant No. KJA202135).

References

[1] H. Zhou, W. Xu, J. Chen and W. Wang, ”Evolutionary V2X Technologies Toward the Internet of Vehicles: Challenges and Opportunities,” *Proc. IEEE.*, vol. 108, no. 2, pp. 308-323, Feb. 2020,

[2] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis and E. K. Markakis, ”A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues,” *IEEE Commun Surveys Tuts.*, vol. 22, no. 2, pp. 1191-1221, Secondquarter 2020,

[3] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane and Y. Liu, ”Multi-UAV-Enabled Load-Balance Mobile-Edge Computing for IoT Networks,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898-6908, Aug. 2020,

[4] J. Zhang, H. Guo, J. Liu and Y. Zhang, ”Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution,” *IEEE Trans.Veh.Technol.*, vol. 69, no. 2, pp. 2092-2104, Feb. 2020,

[5] F. Zeng, Q. Chen, L. Meng and J. Wu, ”Volunteer Assisted Collaborative Offloading and Resource Allocation in Vehicular Edge Computing,” *IEEE Trans.Intell.Transp.syst.*, in press, doi: 10.1109/TITS.2020.2980422.

[6] H. Wu, N. Zheng and B. Chen, ”Feature-specific denoising of neural activity for natural image identification,” *IEEE Trans.Cogni&Deve.Syst.*, in press, doi: 10.1109/TCDS.2021.3062067.

[7] L. Liu et al., ”Neural Human Video Rendering by Learning Dynamic Textures and Rendering-to-Video Translation,” *IEEE Trans.Visuali&Compu.Grap.*, in press, doi: 10.1109/TVCG.2020.2996594.

[8] T. Wang, Y. Luo, J. Liu, R. Chen and K. Li, ”End-to-End Self-Driving Approach Independent of Irrelevant Roadside Objects With Auto-Encoder,” *IEEE Trans.Intell.Transp.syst.*, in press, doi: 10.1109/TITS.2020.3018473.

[9] B. Cao, Z. Sun, J. Zhang and Y. Gu, ”Resource Allocation in 5G IoV Architecture Based on SDN and Fog-Cloud Computing,” *IEEE Trans.Intell.Transp.syst.*, in press, doi: 10.1109/TITS.2020.3048844.

[10] X. Xu et al., ”Service Offloading with Deep Q-Network for Digital Twinning Empowered Internet of Vehicles in Edge Computing,” *IEEE Trans Ind.Informat*, in press, doi: 10.1109/TII.2020.3040180.

[11] W. Sun, H. Zhang, R. Wang and Y. Zhang, ”Reducing Offloading Latency for Digital Twin Edge Networks in 6G,” *IEEE Trans.Veh.Technol.*, vol. 69, no. 10, pp. 12240-12251, Oct. 2020,