Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

MARL based resource allocation scheme leveraging vehicular cloudlet in automotive-industry 5.0

Manzoor Ahmed [a,*], Jinshi Liu [a], Muhammad Ayzed Mirza [b], Wali Ullah Khan [c], Fahd N. Al-Wesabi [d]

[a] College of Computer Science and Technology, Qingdao University, Qingdao 266071, China
[b] BUPT-QMUL EM Theory and Application International Research Lab, Beijing University of Posts and Telecommunications, Beijing 100876, China
[c] Interdisciplinary Center for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg City, Luxembourg
[d] Department of Computer Science, College of Science & Art at Mahayil, King Khalid University, Saudi Arabia & Faculty of Computer and IT, Sana'a University, Yemen

ARTICLE INFO

Article history:
Received 4 July 2022
Revised 22 September 2022
Accepted 13 October 2022
Available online xxxx

Keywords:
Automotive-Industry 5.0
Connected and Autonomous Vehicles (CAV)
Vehicular Cloudlet (VC)
Internet of Vehicles (IoV)
Multi-Agent Reinforcement Learning(MARL)
QMix
Task offloading

ABSTRACT

Automotive-Industry 5.0 will use Beyond Fifth-Generation (B5G) communications to provide robust, abundant computation resources and energy-efficient data sharing among various Intelligent Transportation System (ITS) entities. Based on the vehicle communication network, the Internet of Vehicles (IoV) is created, where vehicles' resources, including processing, storage, sensing, and communication units, can be leveraged to construct Vehicular Cloudlet (VC) to realize resource sharing. As Connected and Autonomous Vehicles (CAV) onboard computing is becoming more potent, VC resources (comprising stationary and moving vehicles' idle resources) seems a promising solution to tackle the incessant computing requirements of vehicles. Furthermore, such spare computing resources can significantly reduce task requests' delay and transmission costs. In order to maximize the utility of task requests in the system under the maximum time constraint, this paper proposes a Secondary Resource Allocation (SRA) mechanism based on a dual time scale. The request service process is regarded as M/M/1 queuing model and considers each task request in the same time slot as an agent. A Partially Observable Markov Decision Process (POMDP) is constructed and combined with the Multi-Agent Reinforcement Learning (MARL) algorithm known as QMix, which exploits the overall vehicle state and queue state to reach effective computing resource allocation decisions. There are two main performance metrics: the system's total utility and task completion rate. Simulation results reveal that the task completion rate is increased by 13%. Furthermore, compared with the deep deterministic policy optimization method, our proposed algorithm can improve the overall utility value by 70% and the task completion rate by 6%.

1. Introduction

Automotive-Industry 5.0 will focus on enabling the intelligent interaction between humans and Connected and Autonomous Vehicles (CAV) leveraging next-generation technologies, including 6G communications (Malik et al., 2021), machine learning tools (Sheraz et al., 2021), edge/fog (Ahmed et al., 2022), vehicular cloudlet computing (Liu et al., 2022), blockchain (Qu et al., 2021), etc. Through Industry 5.0, the automotive sector will further enhance the wireless connections' security, speed, reliability, and connectivity (Khan et al., 2022). Moreover, this industry can enable humans to share data in collaboration with intelligent vehicles and develop robust, reliable CAV applications (Ullah et al., 2020). With the development of smart vehicles and the Internet of Vehicles (IoV), vehicles are transforming from travel tools to intelligent terminals (Zhang et al., 2019). However, the ever-increasing requirements for CAV's Quality of Experience (QoE) (Ning et al., 2019), and the explosive growth of various intelligent applications pose challenges to the implementation of applications (Li et al., 2018; Pu and Carpenter, 2021). Especially the complex computation-intensive applications involving decision-making and real-time resource management.

* Corresponding author.
E-mail addresses: manzoor.achakzai@gmail.com (M. Ahmed), jssmith0716@gmail.com (J. Liu), mamirza@bupt.edu.cn (M.A. Mirza), waliullah.khan@uni.lu (W.U. Khan), falwesabi@kku.edu.sa (F.N. Al-Wesabi).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.10.011
1319-1578/© 2022 The Author(s). Published by Elsevier B.V. on behalf of King Saud University.

Vehicular Ad-hoc Networks (VANET) is considered a valuable platform for Intelligent Transportation Systems (ITS), which enables connectivity for clusters of moving or static vehicles to provide road safety and efficiency applications (Peng et al., 2019; Khan et al., 2022). A robust information network is created through VANET leveraging peer-to-peer ad hoc based communication (i.e., Vehicle to Vehicle (V2V), Vehicle to roadside Infrastructure (V2I), and Vehicle to Everything (V2X)), enabling the vehicle's real-time status via onboard units and eventually communicating to other vehicles/entities. Therefore, VANET is not entirely dependent on RSU or sensors for information dissemination. Indeed VANET is still viewed as one of the challenging forms of wireless communication technologies that complement ITS to improve road safety and non-safety applications (Lin and Deng, 2015). Moreover, with the advent of cloud computing, offloading local resources to a shared pool of resources has been an ideal solution for compute-intensive and memory-intensive applications because vehicles' onboard computing resources are limited to handle such tasks (Tokody et al., 2018). Among the cloud infrastructures, centralized cloud computing is one of the choices having abundant resources. However, it suffers from excessive propagation delays and is unsuitable for delay-sensitive tasks. Therefore, Multi-access Edge Computing (MEC) servers deployed at the eNB/gNB, or RSU, and aggregation point (i.e., 4G Core (EPC) or 5G Core (NGC)), emerge as a promising paradigm to provide low-latency tasks offloading solutions (Deng et al., 2020; Spinelli and Mancuso, 2020). Therefore, multi-access edge computing (MEC) servers deployed at the eNB/gNB, or RSU, and aggregation point (i.e., 4G Core (EPC) or 5G Core (NGC)), emerge as a promising paradigm to provide low-latency tasks offloading solutions. However, it requires huge investment, and the emerging computation-intensive processes may overburden MEC servers (Mahmood et al., 2021). On the other hand, today's intelligent vehicles have increasingly high computing resources. Therefore, they can enable a distributed Vehicular Cloudlet (VC) and facilitate VANET applications as an efficient and economic computation offloading platform (El-Sayed and Chaqfeh, 2019; Zhang et al., 2019).

Unlike centralized and edge clouds, VC infrastructure constitutes a shared collection of underutilized vehicle resources, such as communication, storage, computation, and sensing (Skondras et al., 2019). All such resources of vehicles can be combined to create a cloud network enabling computational resource sharing over the IoV. Moreover, VC can assist the MEC paradigm in overloaded situations by effectively improving computing resources without extra deployment costs.

1.1. Related Works

The resource allocation problem in VC is a basic proposition in networks integrated with MEC. The resource of processors and communication links are two main vital parts. The high mobility of vehicles leads to an unstable offloading environment in vehicular networks and highly affects the computing performance of MEC. Several efforts have been proposed to improve offloading, speed computing, and efficient resource allocation. (Dai et al., 2018) integrated load balancing problem with offloading and formulated a joint MEC server selection, computation resource allocation, and offloading ratio determination problem to maximize the system utility. In the meantime, Xiao et al. (2019) introduced vehicle density-aware MEC cooperation and proposed deep learning-based load prediction and game theory-based MEC grouping for computation offloading. In another work, Zhao et al. (2019) goes a step further by designing a collaborative computation offloading scheme for cloud-assisted MEC in VANETs, where the offloading strategy and resource allocation are jointly optimized. Zhou et al. (2019) focused on considering resource allocation and task assignment problems. The authors proposed an incentive-based model that relies on contract theory to motivate nearby vehicles to share resources. A multiarmed bandit-based distributed adaptive learning algorithm is presented in Sun et al. (2019) with fast varying network topologies to facilitate offloading among vehicles. Moreover, in MEC-aided wireless networks, emerging edge-cloud orchestrated computation offloading (Dai et al., 2020) proposed partial offloading to execute fractional tasks locally and offload the remaining part to edge/cloud servers. Another critical challenge in MEC-based vehicular networks is jointly considering allocating spectrum, computing, and storage. Moreover, owing to the vehicles' high mobility and the network status that changes over time, conventional optimization techniques cannot solve such complex situations. For tackling the issues mentioned above, Peng and Shen (2020) considered the resource allocation problem from two different perspectives, i.e., Macro eNodeB (MeNB) and edge node, both equipped with MEC servers. Furthermore, to simultaneously support multiple offloading tasks and ensure their QoS requirements, the authors extended their work (Peng, 2020) to utilize Unmanned Aerial Vehicles (UAV) in the MEC network to assist in offloading tasks. In addition, the authors proposed an RL-based improved Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm, which can be trained offline to make decisions about resource allocation when executed online. However, the exploitation of dynamic vehicles with idle resources remains unexplored.

Due to the geographical distribution of vehicles, the VC system has a high degree of dynamics and resource volatility. The traditional resource optimization method (Raza et al., 2020) can no longer meet the requirements of its dynamic resource management and allocation. In recent years, due to the rapid development of artificial intelligence technology, researchers have leveraged reinforcement learning to achieve dynamic management of vehicle computing resources (Jameel et al., 2020). For instance, in the work Jiang et al. (2018), the authors leveraged task duplication to solve the non-completed task problem when the vehicle leaves the VC coverage area. They proposed a balanced task assignment strategy and proved it optimal; however, the authors considered the same task size. In another work Sun et al. (2018), the task migration technique is considered in the VC system to minimize the overall response time. They considered tasks with topological order to be migrated to other vehicles before they were completed. In Wang et al. (2018), the authors utilize public vehicles to provide computing services based on the M/M/C priority queue model. Also, a sensory application offloading strategy is proposed based on the semi-Markov decision to obtain the optimal resource allocation plan and maximize long-term rewards. However, the scheme lacks performance in more complex vehicular environments. Another work Lin et al. (2018) considers the scenario of heterogeneous vehicles and RSU computing resource allocation and requests for different types of tasks following different Poisson distributions.

Different from RL, Deep Reinforcement Learning (DRL) technology combines the perception ability of deep learning with the decision-making ability of RL. Moreover, DRL does not need to clarify the probability of state transition and is only based on the current situation, a sample of the system state, and the empirical strategy of objective rewards. The DRL model effectively improves the poor performance of traditional RL algorithms in environments consisting of high-dimensional input state spaces or large action sets. For example, in Ning et al. (2019), a three-layer offloading framework is proposed to minimize overall energy consumption considering moving and stationary vehicles. Due to the high computational complexity, the resource allocation problem was decomposed into traffic redirection and flow. There are two parts to the offloading decision. The Edmonds-Karp algorithm is used for the traffic redirection problem, and the offloading decision part uses the Double Deep Q-Network (H. Van Hasselt et al., 2016) (DDQN). In another work Qi et al. (2019), the authors proposed a knowledge-driven IoV service offloading framework based on the Asynchronous Advantage Actor-Critic (A3C) algorithm that can be trained on multiple different edge nodes at the same time. The task latency is reduced through mobility access to edge computing nodes. Then, the learned knowledge is transferred to the VC controller to make decisions and better adapt to environmental changes. In order to make the vehicle have a real-time response to the vehicle application in the case of resource constraints. In Lee et al. (2020), the authors used Recurrent Neural Network (RNN) to extract time and location from high-dimensional information and continuous behavior space in the environment. The resource availability model uses Proximal Policy Optimization (PPO) algorithm to allocate computing resources and shows higher service satisfaction.

1.2. Motivation and Contributions

Unlike our VC-based resource allocation framework, earlier research essentially considered MEC server-based task offloading (Dai et al., 2018; Xiao et al., 2019; Zhao et al., 2019; Zhou et al., 2019; Sun et al., 2019; Dai et al., 2020; Peng and Shen, 2020; Peng, 2020). As a result, the research deliverables were mainly focused on load balancing, improving offloading, speeding up computing, and resource allocation. In VC based environment, some works (Jiang et al., 2018; Sun et al., 2018; Wang et al., 2018) utilized traditional Reinforcement Learning (RL) algorithms; however, the schemes as mentioned earlier cannot efficiently handle a high degree of dynamics, resource volatility, and complex environmental conditions. Therefore, a practical resource allocation mechanism is required for offloading computational tasks in a VC environment.

Different from (Ning et al., 2019; H. Van Hasselt et al., 2016; Qi et al., 2019; Lee et al., 2020), our paper proposes a model for maximizing system utility value for task offloading and resource allocation problems in the VC environment. First, the impact of computing resource allocation on offloading tasks is quantified by creating the association between system utility value and task completion rate. Eventually, the Secondary Resource Allocation (SRA) mechanism is proposed to deal with the effect of computing resources' fluctuation on task offloading in the VC environment leveraging dual time scale and considering actual scenario parameters, i.e., heterogeneous vehicles and task requests. Furthermore, unlike the existing works based on single-agent reinforcement learning, we opted for multi-agent reinforcement learning (MARL). Then, the SRA mechanism is combined with QMix, i.e., the SRA-QMix algorithm, to find the optimal allocation policy. Our main contributions are summarized as follows:

(1) We investigated the computing resource allocation problem in the VC system. We constructed a model that considers the heterogeneity of tasks and vehicles to maximize the system's overall utility under delay constraints.
(2) Due to the dynamic changes in a multi-agent environment, we decompose the resource allocation problem into two parts based on dual time scales. The two-layered allocation architecture includes large-time and short-time scale actions connected through their respective policy's utility value.
(3) We developed the SRA-QMix algorithm based on our computing allocation model to handle the multi-agent problem, which is an off-policy method. In addition, a mixing network can calculate the total Q value for all the agents.
(4) Simulation results show that the SRA mechanism can adapt well to the VC environment changes. Furthermore, compared with the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) method, our proposed model can improve the utility value under the delay constraints.

The rest of this paper is organized as follows. We first provided an overview of the system model in Section 2. After that, we illustrate the SRA mechanism and the setting of the MARL elements based on our model in Section 3, followed by the introduction of the QMix workflow in Section 4. The following Section 5 is the part for experimental results and discussion. Finally, we conclude this paper and suggest future works in Section 6. The main notations in this paper are shown in Table 1.

2. System model

In real scenarios, vehicles have distinct computing capabilities and hence require diverse computing provisions. Therefore, the resource allocation system should consider heterogeneous vehicles and heterogeneous computing tasks. In addition, for the complex and changeable scenarios of the VC environment, the resource allocation system needs to reasonably allocate the task requests in the case of resource fluctuation. Another point is that the resource allocation system needs to have the ability to globally and uniformly handle multiple vehicles' task allocation without falling into a locally optimal solution. This section first introduces the system architecture, communication model, and queue model and finally gives the optimization objective of the system.

2.1. System Architecture

As shown in Fig. 1, the VC system model primarily comprises the RSU equipment and vehicles. The control center of the VC system is deployed at the RSU and uses V2I communication mode (Sheraz et al., 2021). Unlike previous studies, our VC system considers heterogeneous vehicles, the number of vehicle types is defined as K, and different vehicles can provide different sizes of computing resources. In addition, to better quantify the resource size in the VC, this paper uses the resource unit (RU) to represent the smallest resource unit in the entire VC. The center VC control system allocates the resources available in the resource pool.

The vehicles entering and leaving the VC follow the Poisson distribution, expressed as $\{v_1, v_2, \ldots, v_C\}$, and C is the number of vehicles in the VC. Due to the limited computing resources of the vehicle, when the vehicle generates a task request, it will be sent

Table 1
List of Key Notations.

Symbol	Description
k, j	*No. of vehicle type and task type*
C	*No. of vehicles in RSU coverage*
H	*No. of time slots(steps) in a episode*
α, β	*The corresponding relation of RU number and vehicle type, penalty coefficient*
p_j	*The probability of task type-j*
d_j,t_j^{max},ϕ_j	*The task size, maximum delay and utility parameter of task-j*
r_c	*The transmission rate between vehicle C and RSU*
N_k	*Total number of type-k vehicles in VC*
N_i^{ru}	*No. of RU assign to task i*
λ, μ	*The vehicles arrival rate and service rate in VC*
μ_u	*Service rate for one RU*
P_n^{tr}, B	*Transmission power and spectral bandwidth*
σ, h_t	*Gaussian noise power and channel attenuation coefficient*
d,v	*Distance and path loss exponent*
X_{ij}, Y_{ij}	*The indicator for execute task in VC or edge server*
e_{ar}, e_d	*Arrival and department event*
k_q	*The task q in the queue line*
$Q_{tot}^{\pi}(s_t, u_t)$	*The total Q value for all agents*

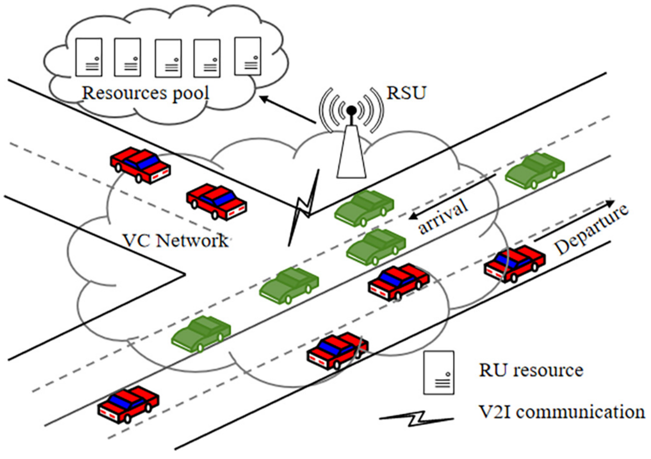

Fig. 1. The VC architecture.

to the VC control center. Once the VC receives the task request, it decides whether to accept or reject it based on the task information and the number of resources in the current resource pool. If enough resources are available, the VC allocates a corresponding number of RUs; otherwise, the VC rejects the request and sends the service request to the nearby edge server Raza et al. (2019) for execution. In this case, the system will be punished.

In order to better fit the actual situation, different task requests have different descriptive characteristics. For example, a strict time limit is required for automatic navigation tasks, while computationally intensive tasks such as AR and VR in-vehicle entertainment do not require a strict limit. We define task information as follows:

$$T_j = \{d_j, t_j^{max}, \phi_j\}, j \in J, \tag{1}$$

where J represents the type of task request, d_j and t_j^{max} represent the request size and the maximum time delay of the type-j task, respectively. ϕ_j is a parameter of type-j task, which represents the task utility and can be shown as $\phi_j(t_j^{max} - t_j^{tot})$, where t_j^{tot} is the total time to execute the task. In addition, the generated probability of a type-j task is p_j, and given as $\sum_{j \in J} p_j = 1$.

2.2. Communication Model

The transmission time t_i^{up} of the task request i for the vehicle c is related to the channel transmission rate. The channel transmission rate can be expressed as:

$$r_n^t = B\log_2(1 + \frac{P_n^{tr}|h_t|^2}{\sigma^2 (s_n^t)^{\upsilon}}), \tag{2}$$

where $s^{-\upsilon}$ represents the path loss, s and υ are the distances between vehicle and RSU and path loss exponent, respectively. P_n^{tr} represents the transmission power between vehicle and RSU, and h_t is the channel attenuation coefficient. σ^2 is the Gaussian noise power, and B represents the available spectral bandwidth. At this time, the task transmission time for the task type-j of the vehicle c is:

$$t_j^{up} = \frac{d_j}{r_c} \tag{3}$$

2.3. Queue Model

When a vehicle in the VC system generates a task request, the VC allocates RU resources to serve it. Although multiple task requests are to be served simultaneously, the entire VC can be regarded as a queueing model based on M/M/1. The task waiting for flow in the queue is λ, which is the arrival rate of the task request queue. We assume that the service rate of each RU resource is μ_u, then the service rate of the entire system can be shown as:

$$\mu = \sum_{k \in K} N_k \alpha k \mu_u, \tag{4}$$

where α represents the relationship between the vehicle and the number of RU available and αk represents the number of RU offered by the vehicle type-k. N_k represents the total number of type-k vehicles in the VC coverage and meets $\lambda < \mu$. According to the queue theory, the waiting time for a request task i in the queue is:

$$t_i^{wat} = \frac{\lambda}{\mu(\mu - \lambda)}. \tag{5}$$

In addition, the execution time of the task i can be expressed as:

$$t_i^{pro} = \frac{1}{\mu_u N_i^{ru}}, \tag{6}$$

where N_i^{ru} is the number of RU assigned to the task i.

2.4. Optimization Objective

In a given time slot h, for a type-j request generated by a vehicle c within the coverage of the VC, use $X_{ij}^h = 1$ to indicate that the VC accepts the request, and use $Y_{ij}^h = 1$ to indicate that the VC sends the request to a nearby edge server. In other cases, X_{ij}^h and Y_{ij}^h are both 0. The optimization goal of this system is to adjust the resource allocation of vehicles' task requests within the scope of VC in a given time slot, and maximize the system's utility under the limitation of task delay requirements, thereby improving the quality of user service. The optimization problem can be written as:

$$\begin{aligned} \max_{XYN^{ru}} U = & \sum_{h=1}^{H}\sum_{j=1}^{J}\sum_{i=1}^{N} X_{ij}^h p_j \phi_j \left(t_{ijh}^{\max} - t_{ijh}^{\text{tot}}\right) - Y_{ij}^h \beta \\ s.t. \quad & X_{ij}^h = \{0,1\}, Y_{ij}^h = \{0,1\} \\ & X_{ij}^h Y_{ij}^h = 0, X_{ij}^h + Y_{ij}^h = 1 \\ & \sum_{j \in J} p_j = 1 \\ & 0 < N^{ru} < L. \end{aligned} \tag{7}$$

Where β is the penalty when the VC chooses to send the task to the nearby edge server. H is the number of time slots in one training episode, t_i^{tot} is total time for finishing task-i, which can expressed as $t_i^{tot} = t_i^{up} + t_i^{wat} + t_i^{pro}$. The first two constraints mean that VC can only choose one processing method for one request task, and L represents the maximum number of RUs that can be allocated at one time.

3. SRA and RL Elements

This section aims to explain the working of a secondary resource allocation scheme abbreviated as SRA based on multi-agent reinforcement learning elements, i.e., state, action, and reward in a dynamic VC environment. Basically, our goal is to maximize the system's utility conditional to task delay requirements, thereby improving the quality of user service as already mentioned in 7. SRA mechanism is based on the dual time scale, which can adapt to the high-speed dynamic environment. Through the Partially Observable Markov Decision Process (POMDP), agents' state, action, and cumulative rewards are defined, which learn to collaborate in a random environment to maximize overall rewards.

3.1. SRA Mechanism

Due to the dynamic nature of the VC environment, vehicles' arrival or departure will affect the VC's resource availability, which in turn affects the resource allocation decision. Unfortunately, the existing offloading methods cannot meet the computing requirements of the task dynamically. Therefore, our proposed approach uses the SRA mechanism based on the dual time-scale model.

In our model, two kinds of resource allocation actions are considered, i.e., large-time-scale and small-time-scale, as shown in Fig. 2. The decision is made through a large-time-scale action, indicated by a light green circle in the figure. Moreover, the solid black circles in the bottom line represent the system time frame, which is the time of one step in the simulation environment. It can be seen that the interval of large-time-scale actions is not fixed because the interval of receiving task requests is different. In addition, the large-time-scale action interval is relatively long; the VC environment state may change during this period. For example, if a task is completed in advance, the corresponding RU resources can be allocated again. Alternatively, a task cannot be completed if the vehicle leaves the VC due to a change in the vehicle speed, leading to task completion failure. If the allocated RU resources can be adjusted according to the changes in the environment during the large time-scale action interval, the task completion rate can be significantly improved. This adjustment action is defined as a small-time-scale action and is represented by a blue diamond in the figure. It can be seen that the action interval of the small-time scale is fixed and relatively short so that the available resources of the system can be continuously adjusted. If a task requests to allocate sufficient resources, then the number of RUs can be appropriately reduced. On the contrary, for tasks that cannot be completed within the maximum time constraint, in such a scenario, the allocated RUs can be increased through the resource pool so the task completion time can be reduced. In general, whether the task request is accepted or not by the large-time-scale action is determined, and the number of allocated RUs is continuously adjusted by the small-time-scale action to track subtle changes in the environment. This way, the model can be well adapted to the high-speed dynamic VC environment through the SRA mechanism.

3.2. Data Processing as MDP

Unlike single-agent reinforcement learning, in MARL, as the state space becomes larger, the joint action space increases exponentially with the number of agents, so it is difficult to fit an appropriate function to represent the actual joint action-value function. The agent must learn to collaborate under limited computing resources in such a random environment to maximize overall rewards. This article uses tuple $G = (S, U, P, R, O, \gamma)$ to describe the POMDP, where $s \in S$ represents global environmental information. In each time slot, an action $u^a \in U$ needs to be selected for an agent $a \in A$ to form a joint action U. By applying the joint action to the environment, enter the next state according to the state transition probability $P(s\prime|s, u)$. At the same time, all agents will receive a reward of $r(s, u)$. In the actual observation process, the agent can only obtain its state information, and different agents have different observation information, represented by $O(s, a)$. Therefore, the RNN can achieve better results for incomplete observations, and τ^a is used to represent the action observation history of agent a, and the strategy function $\pi^a(u^a|\tau^a)$ is constructed based on this action observation history so that the joint action-value function can be obtained:

$$Q^{\pi}_{tot}(s_t, u_t) = \mathbb{E}_{s_{t+1},u_{t+1}}\left[\sum_{i=0}^{H}\gamma^i r_{t+i}|s_t, u_t\right], \tag{8}$$

where γ is the discount factor.

3.2.1. State and Observation

For global state s_t in the VC environment, all information such as vehicles and queues needs to be considered, which can be expressed in the following form:

$$S = \{v_1, v_2, \ldots, v_C; k_1, k_2, \ldots, k_q; p_1, p_2, \ldots, p_n; e \in \{e_{ar}, e_d\}\} \tag{9}$$

where v_c represents the information of vehicles within the VC range, including vehicle type, location, speed, and task requests generated. k_q represents the request information waiting in the queue, p_n represents the request being executed, and e represents the event in the current time slot, including the request arrival e_{ar} and department e_d. The partial observation information o_a that needs to be obtained for the task request with the current time slot is:

$$O_a = \{d_a, t^{\max}_a, \phi_a, k_a, v_a, g_a\}, \tag{10}$$

where k_a represents the vehicle type of request a, the different vehicles can offer RU numbers. v_a and g_a represent the speed and position coordinates of the vehicle in the current time slot, respectively.

3.2.2. Action

Since the system model uses the SRA mechanism based on dual time scales, there are two types of actions. For large time-scales, the joint action at time slot t is $U_l = \{a_1, a_2, \cdots, a_n | a \in \{0, 1, \ldots, l\}\}$, where a_n represents the number of RU resources allocated to the n-th request, and the maximum value is l. When $a_n = 0$ represents, the VC rejects and sends the request to a nearby edge server. For small time scale, the joint action in time slot t is $U_s = \{a_1, a_2, \cdots, a_n | a \in \{-s, 0, s\}$, where $-s$ and s indicate the number of RUs that are reduced or increased the number of s for a certain request, and when $a_n = 0$ indicates that no operation is performed for the request task.

3.2.3. Reward Function

The optimization goal of this system is to maximize the system's overall utility, and the goal of reinforcement learning is to maximize the cumulative rewards obtained, so the reward function value is defined as the utility value obtained in each time slot. For the task execution in the vehicle cloud environment, the delay of task execution is critical to the user experience. Therefore, the optimization goal of this paper is to minimize the total time t^{tot} of each task request. Therefore, for the utility of each task request, define $\Delta t = (t^{max} - t^{tot})$ as the difference between the total request completion time t^{tot} and its maximum time t^{max} limit, and the reward function is as follows:

$$R(T_i) = \begin{cases} \Delta t \phi_j + \eta N^{ru}, & \Delta t \geqslant 0 \\ \frac{\Delta t}{\rho t^{max}} \phi_j, & -\rho t^{\max} < \Delta t < 0 \\ \beta, & \text{otherwise} \end{cases} \tag{11}$$

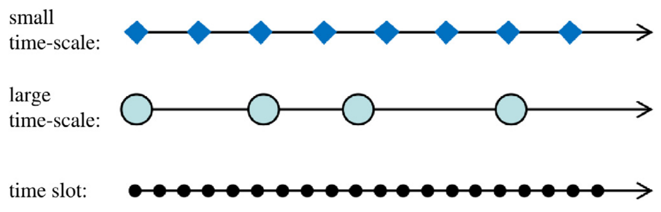

Fig. 2. Illustration of dual time-scale of SRA mechanism.

Where β represents the penalty when the control center rejects the requested task according to the current state, η represents the utility value for one RU, and ρ is a super parameter that meets $0 < \rho < 1$. For the first case, $\Delta t \geqslant 0$, the time required to complete the current task is less than the maximum delay, leading to higher user service satisfaction. In this case, a positive reward should be given to encourage the system to allocate more RU to reduce the execution time. For the second case $-\rho t^{\max} < \Delta t < 0$, it means that the task is not completed within the time limit, but the exceeded time is still within a certain range. In this case, the system can still get some reward, and the range is controlled by the parameter ρ. For example, if VC receives a task having size d is 60, the maximum time $t^{\max}$ is 15s and the parameter ϕ is 3. If the control center rejects this request, the utility value of this task is $-\beta$, and the value is -10. If the system receives the task and allocates 5 RU resources, the execution time is $60/5 = 12s$, and Δt is 3s. At this time, the utility value of the task can be obtained according to the Eq. (11), which is $3 \ast 3 + 1 \ast 5 = 14$. The optimal resource allocation strategy π is to maximize the overall reward, which can be expressed as:

$$\pi^* = \arg\max_{\pi} \sum_{l=1}^{H} \sum_{i=1}^{N} R(T_{i,l})/H \tag{12}$$

4. QMix-based Resource Allocation Scheme

In this section, we will first introduce a Deep Q-Network (DQN) comprising of estimation and target networks to handle the vehicles' continuous state environment. After that, QMix training is focused, combined with the SRA mechanism named SRA-QMix algorithm. In this algorithm, each agent adopts actions based on a greedy strategy through Deep Recurrent Q-Learning (DRQN) structure.

4.1. Deep Q-Network

DQN introduces a neural network based on the Q-Learning algorithm to replace the Q table so that it can handle a continuous state environment, and its loss function is:

$$L(\theta) = \mathbb{E}\left[(\hat{y}_\theta - q(s, a, \theta))^2\right], \tag{13}$$

where $\hat{y}_\theta = r + \gamma \max_{a_{t+1}} \hat{q}(s_{t+1}, a_{t+1}, \theta^-)$. There are two neural networks in the DQN algorithm. One network is an estimation network $q(s_{t+1}, a_{t+1}, \theta)$ with parameter θ, which will be updated each time when the current state value is calculated. The other one is target network $\hat{q}(s_{t+1}, a_{t+1}, \theta^-)$ with parameter θ^-, which is used to calculate the action value of next state. In the training process, by setting the update frequency of the target network to be lower than the estimation network, the target network parameters remain unchanged for some time. After that, the target network updates the parameters from the estimation network, which can improve the stability and convergence speed of the algorithm.

4.2. QMix Training

By combining the QMix algorithm with the SRA mechanism, this paper proposes a new computing resource allocation algorithm SRA-QMix, which uses the Centralised Training with DEcentralised (CTDE) mechanism. As shown in Fig. 3, the agent represents the request to be processed, and the sum of the partial observations of all agents in the global state, that is, the Q network is trained through the global state s_t information to obtain the global value function $Q_{tot}(\tau, u)$. In the decentralized execution stage,

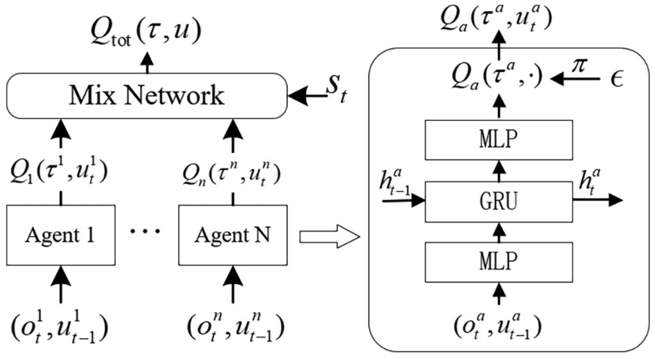

Fig. 3. The process of calculating global Q_{tot} in the QMix algorithm.

the action value is estimated according to the request information of each task, the local observation state o_i of the vehicle to which it belongs, and each agent adopts a greedy strategy to select actions.

Algorithm 1: SRA-QMix

```
   Input: The global state $s_t$ and observation state $o_s$
          for all the agents.
   Output: Union action $u$ with maximum utility value.
 1 Initialize the parameters of $\lambda$, $\mu_u$, $\beta$, $\eta$, $\rho$
 2 Initialize neural network weight $\theta, \theta^-$ and super
     parameters
 3 foreach episodei in N do
 4     foreach step $t = 1, 2, \ldots, H$ do
 5         if $random < \epsilon$ then
 6             With probability $\epsilon$ select a action $a_i$ for
                 each agents
 7         else
 8             $a = arg\max(Q(s, u; \theta))$ for each agents
 9         end Execute union action $u_t$
10         Get reward $r_t$ and next state $s_{t+1}$
11     end
12     foreach step t in [large time, small time] do
13         Store the transition sequence into buffer $B$
14     end
15     if buffer size large than batch size then
16         Sample random transition sequence from $B$
17         Calculate $q$ and $\hat{y}$:
             $\hat{y} = r_{t+1} + \gamma \max_s Q(s_{t+1}, a; \theta^-)$ for each
             agents
18         Get $Q_{tot}(\tau, u)$ from (16) and $\hat{y}^{tot}$ in (18)
19         Calculate loss from (17)
20         Execute step (16,17,18,19) for large time and
             small time.
21     end
22     After $f$ step, update the target network
         parameters with $\sigma$: $\theta^- = \sigma\theta + (1 - \sigma)\theta^-$
23 end
```

The monotonicity of the global value function $Q_{tot}(\tau, u)$ and the single agent value function $Q_n(\tau^n, u^n)$ should be the same, then the operation of argmax on the global value function and the operation of argmax on the value function of a single agent can get the same result:

$$\arg\max_{u} Q_{tot}(\tau,u) = \begin{pmatrix} \arg\max_{u_1} Q_1(\tau_1,u_1) \\ \vdots \\ \arg\max_{u_n} Q_n(\tau_n,u_n) \end{pmatrix} \tag{14}$$

In order to ensure that the monotonicity of the value function makes the above Eq. (14) hold, the algorithm has the following constraints:

$$\frac{\partial Q_{tot}}{\partial Q_i} \geqslant 0, \forall i \in \{1,2,\cdots,n\} \tag{15}$$

Different from the simple addition of the value functions of all agents in the value-decomposition networks (VDN) algorithm, as shown in Fig. 3, QMix uses a mixing network to integrate the local value functions of a single agent in a non-linear manner:

$$Q_{tot}(\tau,u) = M(s_t, Q_1(\tau_1,u_1),\ldots,Q_n(\tau_n,u_n)), \tag{16}$$

where M represents the mixing network, a forward propagation neural network. In order to meet the requirements of Eq. (15), the parameters of the hybrid network are generated by a separate hyper-parameter network, which adopts a hyper-network structure. The input includes the local value function of each agent and the global state information s_t, and the output is the weight and offset of the hybrid network. In this way, by decomposing $Q_{tot}(\tau,u)$, the amount of calculation to operate on it no longer increases exponentially with the joint action space. However, it linearly increases with the number of agents, improving the algorithm's efficiency. The structure of each agent is DRQN, and the input is the trajectory information observed by a single task. After passing through the GRU recurrent network, it can learn long-term sequences. All agents can use one-hot encoding to share the same network in the program.

SRA-QMix is based on the DQN algorithm, and the basic process is similar to DQN. Both use experience pools to store experience information, and there is also a corresponding target network. The final loss function can be expressed as:

$$L(\theta) = \sum_{i=1}^{b}\left[\left(\hat{y}_i^{\mathrm{tot}} - Q_{\mathrm{tot}}(\tau,u,s;\theta)\right)^2\right] \tag{17}$$

where b is the number of samples sampled from the experience pool and $\hat{y}_i^{tot}$ is:

$$\hat{y}_i^{\mathrm{tot}} = r + \gamma \max_{u_{t+1}} \hat{q}(\tau_{t+1},u_{t+1},s_{t+1};\theta^-) \tag{18}$$

5. Simulation results and Discussion

5.1. Simulation Environment

The simulation software uses SUMO, the hardware uses Intel Xeon W-2133, 32 GB memory, based on Ubuntu 18.04, and the simulation program uses Python 3.6 + Pytorch 1.8. The simulation scenario considers an area of an intersection, deploys an RSU, and serves as the control center of the VC. After the vehicles enter the VC environment, it is assumed that they will share computing resources. At the same time, each vehicle has a certain probability of generating task requests. The algorithm adopts a degradation strategy, and some simulation parameters are shown in Table 2.

In order to verify the performance of the computing resource allocation algorithm SRA-QMix and the effectiveness of its dual time scales, this paper considers the following three algorithms as benchmarks for comparison:

Table 2
Simulation Parameters.

Parameter	*Value*
The length of one time slot t	0.2s
No. of step in a episode H	300
The parameter of α and ρ	1–2, 0.2
The penalty coefficient β,	−10
The utility value for one RU η	1
The task size for all type d_j	60,90,120
The task maximum for all type t^{max}	15/16/19
The utility coefficient for all task type ϕ_j	3,2,1
The speed of vehicle	8–12 m/s
The learning rate, buffer size for QMix lr,B	4e-4, 1e4
The learning rate, buffer size for MADDPG	4e-4, 1e4

- QMix (Rashid et al., 2020): QMix estimates the joint Q value function as a complex nonlinear combination of each agent's local value functions, improving algorithm performance. Moreover, it has better performance in multi-agent environments. This baseline algorithm did not use the SRA mechanism for QMix, and parameter settings are consistent with SRA-QMix.
- MADDPG (Lowe et al., 2017): the multi-agent improved version of the deep deterministic policy gradient (DDPG). It enables distributed execution of centralized training and allows the Critic to obtain all agent observations during the training period.
- SRA-MADDPG: In this baseline, the SRA mechanism is with the MADDPG scheme. This baseline is added to verify the effectiveness of the SRA mechanism in improving algorithm performance.

There are two main performance metrics in this paper, i.e., the system's total utility and task completion rate. The total utility value of the system is calculated according to the reward function. The optimal strategy in the RL algorithm is to obtain a higher utility value. The higher the utility value means, the better system's decision-making and the more reasonable allocation of resources to get the maximum reward. Another metric is the task completion rate, which indicates a ratio of the number of successfully completed tasks to the total number of offloading tasks. In the VC environment, if the distance between the task vehicle and the service vehicle exceeds the network coverage radius and the task has not been completed, the task execution will fail. The higher the task completion rate, the more tasks are completed in the vehicle network, improving resource utilization and user satisfaction.

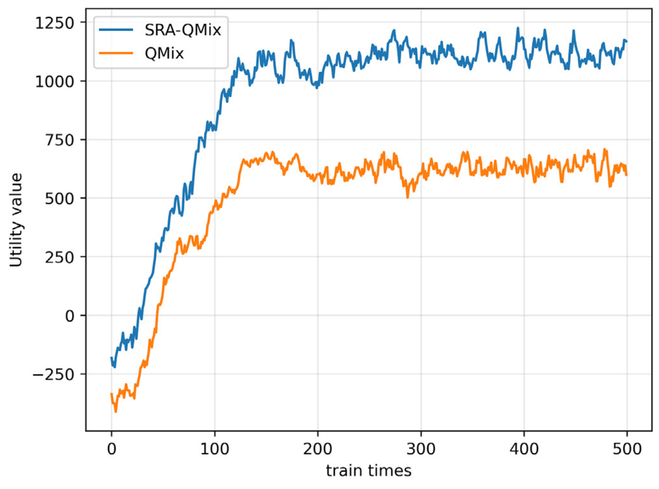

Fig. 4. The training process.

5.2. Performance Evaluation

Fig. 4 shows the total utility value versus the training time. For instance, when the vehicle arrival rate is 0.9 and parameter α is 1.5, the SRA-QMix algorithm has a higher utility value than the QMix algorithm. As the number of iterations increases, the curves of the two algorithms reach a plateau around iteration 150. After stabilization, SRA-QMix is 70% higher than QMix. The reason is that SRA-QMix applies an SRA mechanism, which can better adapt to changes in the environment, thereby ensuring the successful completion of task requests, obtaining higher system utility values, and effectively improving the utilization of computing resources.

Fig. 5 shows the variation of the total utility value under different algorithms with the vehicle arrival rate λ. The larger the λ is, the more vehicles enter the VC environment simultaneously, and the number of requests that need to be processed in the same time slot will increase accordingly. When λ is 0.3, the utility values of the three algorithms are relatively low because the number of tasks in the system is relatively small, so the overall utility value is relatively low. Then with the increase of λ, the overall utility value of the three algorithms increases. It can be seen that the SRA-QMix algorithm model in this paper can obtain a higher overall utility value. It shows that the algorithm can allocate computing resources more reasonably and deal with the cooperative and competitive relationship between multiple tasks so that the decision can maximize the overall utility. When the value of λ exceeds 1.7, the growth rate of the utility value of the three schemes begins to decrease, and the curve becomes smooth. This is because the computing resources allocated in the VC are limited, and the number of task requests reaches the maximum utility value that the system can obtain under the current resource conditions. Hence, the overall utility value increases slowly. Moreover, the SRA mechanism can adapt well to the complex vehicle multi-agent environment compared with other algorithms.

Fig. 6 shows the relationship between task finish rate and vehicle arrival rate λ under different algorithms. The task completion rate is the ratio of the number of tasks completed in VC to the total number of requests received. A higher value of the task completion rate indicates that more tasks can be completed within the VC instead of being sent to a nearby edge server for execution, thus improving user service satisfaction. It can be seen that as the arrival rate of vehicles increases, the finish rate of task requests decreases. For example, at λ=0.3, the task finish rate of the SRA-QMix algorithm can be close to 90%, improved by 5% compared with the SRA-MADDPG algorithm. On the other hand, as the vehicle arrival rate increases, the task completion rate of the four algorithms decreases. Also, with the increase in the number of tasks, the waiting time of tasks in the queue increases, and the total time to complete task requests increases. Failure to complete the mission before the vehicle leaves will result in mission failure. The reason is that the limited RU resources cannot meet the computing requirements. The algorithm actively sends the task to a nearby edge server for execution according to the current queue situation and the number of RUs, decreasing the task service rate. In the case of limited RU computing resources, the algorithm's allocation strategy is no longer accurate, and tasks are completed before the maximum time limit. As a result, the probability is reduced, and the QMix algorithm with the SRA mechanism outperforms the baseline schemes.

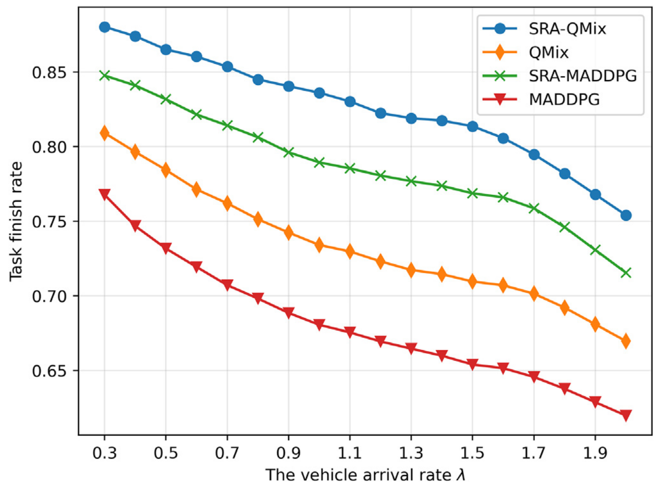

Fig. 6. The task finish rate with various parameter λ.

From Figs. 4 and 5, the performance of the algorithm MADDPG is weaker than that of QMix because MADDPG is improved based on the deterministic strategy gradient algorithm. For the Critic, the global state can be obtained for guidance in training a single Actor. When testing, the Actor takes actions based on local observations and follows the centralized training and distributed execution process. However, due to the lack of QMix's Q-value mix mechanism for each agent, there is no overall reward function, and the collective performance of agents under resource constraints is weaker than the QMix algorithm.

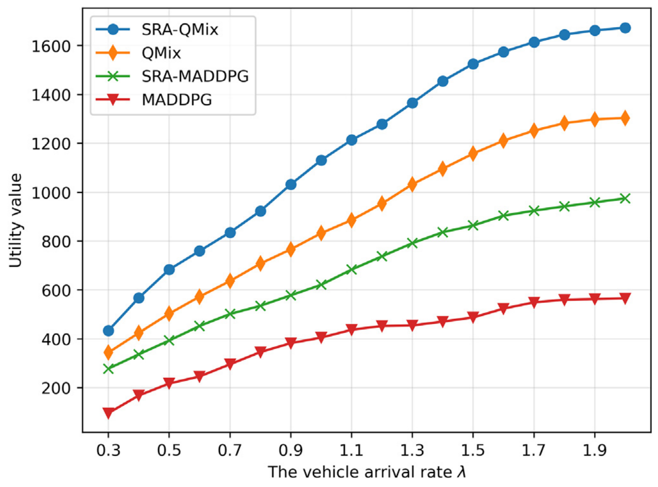

Fig. 5. The utility value with various vehicle arrival rates λ.

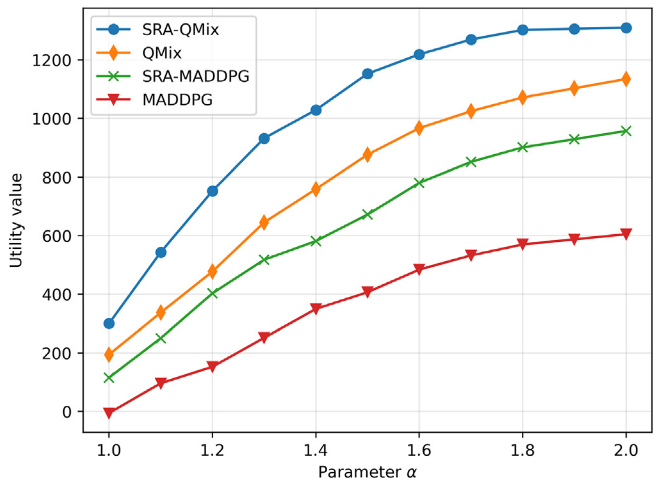

Fig. 7. The task finish rate with various values of α.

Fig. 7 shows the relationship between the task system utility with the parameter α. Among them, α represents the relationship between the vehicle type and the number of RUs. The larger the α, the more computing resources the same vehicle can provide, and the overall number of RUs will increase. We set the number of requests that can be processed each moment to 10. In the beginning, the value of α is 1.0, the amount of RU provided by the vehicle is minimal, and it can be seen that the total utility is relatively low. Because the amount of RU allocated by the VC control center at this time is minimal, the scheduling space is limited, which cannot meet the computing requirements of vehicle task requests. As the value increases, the same vehicle can contribute more RUs, and the control center can have enough space to reasonably allocate these RU computing resources. More resources mean requests can be completed quickly, thus improving the total utility. The model uses the SRA mechanism, an improvement over the original model. It can get the same conclusion from Fig. 8 that shows the between the task finish rate with the parameter α. When the value of parameter α increases, the task finish rate increases. Then, if the value of α exceeds 1.8, the curve becomes smooth. Because in this case, the number of RUs is sufficient relative to the number of task requests so that the requests can be completed quickly. Thus, the curve increases slowly after the maximum utility obtained from the current number of tasks is reached. One conclusion is that using the SRA mechanism can increase system performance.

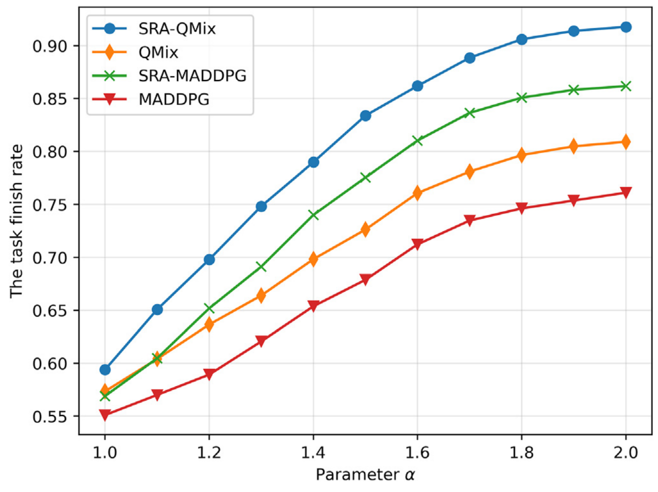

Fig. 8. The task finish rate with various values of α.

This paper proposed the SRA mechanism, which adjusts the allocated resources with a small time-scale action frequency to adapt to the resource volatility of the VC environment. Fig. 9 shows the small-time scale's total utility and task finish rate at different update intervals. The performance shown here is when the vehicle arrival rate is λ=1 and α=1.5. The smaller the update interval of the small-time scale, the more frequent the update. When the update interval is 1, the number of allocated RUs will be adjusted through small-time scale updates at each environment step. From Fig. 9, when the interval is 1, both algorithms have the highest utility value and task finish rate. The total system utility and task finish rate decrease as the update interval increases. It shows that the high-frequency small-time scale update can adapt to the highly dynamic VC environment and help the algorithm learn better resource allocation strategies.

5.3. Discussions

For the problem of task offloading and resource allocation in the VC environment, this paper does not consider the influence of surrounding traffic flow on allocation strategy. However, adding traffic flow information to the model will help the algorithm make better decisions. Specifically, suppose the traffic flow is predicted to increase in the future. In that case, a conservative allocation strategy can be implemented at this time to reserve some computing resources in advance to deal with a large number of task requests to improve the task completion rate. Conversely, if traffic flows decrease in the future, aggressive strategies can be used to allocate more resources to reduce task execution time to improve users' quality of experience.

In addition, the privacy and security of users need to be considered in depth to ensure that the privacy of vehicle users sharing resources will not be threatened. With increasing attention, DRL technology has been applied to many fields and achieved good results; then, how to apply the DRL technology to the general modeling of the task offloading environment of the IoV, including the setting of vehicle status and reward function. So that the combination with the latest DRL algorithm becomes simple and easy to use, and this direction needs to be further investigated.

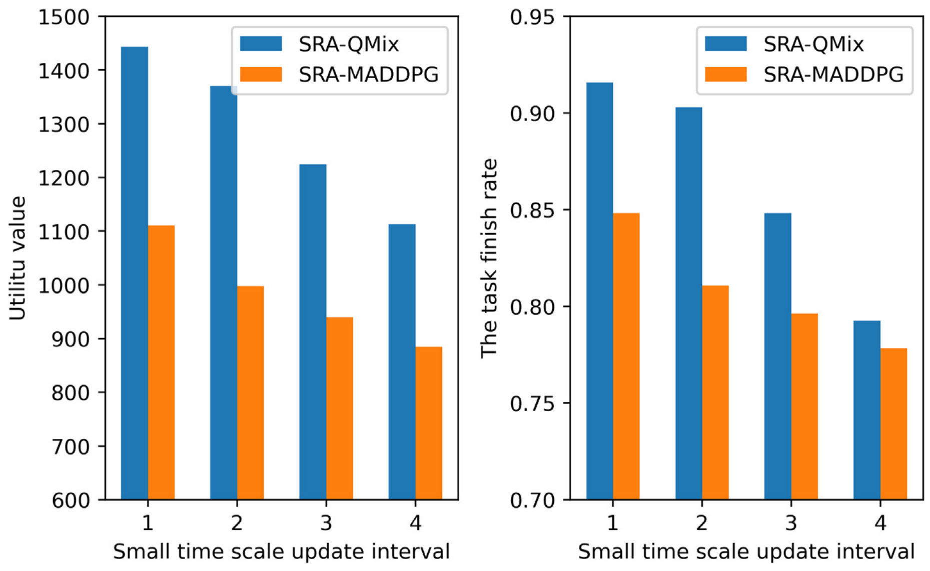

Fig. 9. The utility value and task finish rate with various small time scale update intervals.

6. Conclusion

Aiming at the problem of task offloading and resource allocation in the VC environment, this paper first proposed a model of maximizing system utility value. It then quantified the impact of computing resource allocation on offloading tasks by constructing the relationship between system utility value and task completion rate. Finally, the SRA mechanism is proposed to deal with the impact of the fluctuation of total available computing resources on task offloading in the VC environment. This paper proposed the SRA-QMIX algorithm based on MARL to find the optimal allocation policy for solving the dimensional curse problem in multi-agent. Compared with the deep deterministic policy gradient algorithm, the SRA-QMix can significantly improve the system utility value and task completion rate and better adapt to the VC dynamic offloading environment. In our future work, we would like to extend our research by considering 5G NR and mmWave-based Vehicular Communication Networks (VCN) to evaluate and reduce latency. Moreover, apart from VCN, other critical performance metrics like energy consumption and computation overhead will be analyzed.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmed, M., Raza, S., Mirza, M.A., Aziz, A., Khan, M.A., Khan, W.U., Li, J., Han, Z., 2022. A survey on vehicular task offloading: Classification, issues, and challenges. J. King Saud University - Computer Inform. Sci.

Dai, Y., Xu, D., Maharjan, S., Zhang, Y., 2018. Joint load balancing and offloading in vehicular edge computing and networks. IEEE Internet Things J. 6 (3), 4377–4387.

Dai, Y., Zhang, K., Maharjan, S., Zhang, Y., 2020. Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond. IEEE Transactions on Vehicular Technology 69 (10), pp. 12 175–12 186.

Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y., 2020. Edge intelligence: the confluence of edge computing and artificial intelligence. IEEE Internet Things J. 7 (8), 7457–7469.

El-Sayed, H., Chaqfeh, M., 2019. Exploiting mobile edge computing for enhancing vehicular applications in smart cities. Sensors 19 (5), 1073.

H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in Proceedings of the AAAI conference on artificial intelligence, vol. 30, no. 1, 2016.

Jameel, F., Javaid, U., Khan, W.U., Aman, M.N., Pervaiz, H., Jäntti, R., 2020. Reinforcement learning in blockchain-enabled IIoT networks: A survey of recent advances and open challenges. Sustainability 12 (12), 5161.

Jiang, Z., Zhou, S., Guo, X., Niu, Z., 2018. Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis, and implications on road traffic. IEEE Internet Things J. 5 (1), 93–107.

Khan, W.U., Jamshed, M.A., Lagunas, E., Chatzinotas, S., Li, X., Ottersten, B., 2022. Energy efficiency optimization for backscatter enhanced NOMA cooperative V2X communications under imperfect CSI. IEEE Trans. Intell. Transp. Syst.

Khan, W.U., Ihsan, A., Nguyen, T.N., Javed, M.A., Ali, Z., 2022. NOMA-enabled backscatter communications for green transportation in automotive-Industry 5.0. IEEE Trans. Industr. Inf.

Lee, S.-S., Lee, S., 2020. Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. IEEE Internet Things J. 7 (10), pp. 10 450–10 464.

Li, B., Fei, Z., Chu, Z., Zhang, Y., 2018. Secure transmission for heterogeneous cellular networks with wireless information and power transfer. IEEE Syst. J. 12 (4), 3755–3766.

Lin, C.-C., Deng, D.-J., 2015. Optimal two-lane placement for hybrid vanet-sensor networks. IEEE Trans. Industr. Electron. 62 (12), 7883–7891.

Lin, C., Deng, D., Yao, C., 2018. Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units. IEEE Internet Things J. 5 (5), 3692–3700.

Liu, J., Ahmed, M., Mirza, M.A., Khan, W.U., Xu, D., Li, J., Aziz, A., Han, Z., 2022. Rl/drl meets vehicular task offloading using edge and vehicular cloudlet: A survey. IEEE Internet Things J. 9 (11), 8315–8338.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments, vol. 30.

Mahmood, A., Hong, Y., Ehsan, M.K., Mumtaz, S., 2021. Optimal resource allocation and task segmentation in IoT enabled mobile edge cloud. IEEE Transactions on Vehicular Technology 70 (12), pp. 13 294–13 303.

Malik, U.M., Javed, M.A., Zeadally, S., Islam, S., 2021. Energy efficient fog computing for 6g enabled massive iot: Recent trends and future opportunities. IEEE Internet Things J., 1

Ning, Z., Dong, P., Wang, X., Guo, L., Rodrigues, J.J., Kong, X., Huang, J., Kwok, R.Y., 2019. Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme. IEEE Transactions on Cognitive Communications and Networking 5 (4), 1060–1072.

Ning, Z., Huang, J., Wang, X., 2019. Vehicular fog computing: Enabling real-time traffic management for smart cities. IEEE Wirel. Commun. 26 (1), 87–93.

Peng, S.X., 2020. Haixia, "Multi-agent reinforcement learning based resource management in mec-and uav-assisted vehicular networks,". IEEE J. Sel. Areas Commun. 39 (1), 131–141.

Peng, H., Shen, X., 2020. Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks. IEEE Trans. Network Sci. Eng. 7 (4), 2416–2428.

Peng, H., Liang, L., Shen, X., Li, G.Y., 2019. Vehicular communications: A network layer perspective. IEEE Trans. Veh. Technol. 68 (2), 1064–1078.

Pu, C., Carpenter, L., 2021. Psched: A priority-based service scheduling scheme for the internet of drones. IEEE Syst. J. 15, 4230–4239.

Qi, Q., Wang, J., Ma, Z., Sun, H., Cao, Y., Zhang, L., Liao, J., 2019. Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach. IEEE Trans. Veh. Technol. 68 (5), 4192–4203.

Qu, Y., Pokhrel, S.R., Garg, S., Gao, L., Xiang, Y., 2021. A blockchained federated learning framework for cognitive computing in industry 4.0 networks. IEEE Trans. Industr. Inf. 17, 2964–2973.

Rashid, T., Samvelyan, M., Witt, C.S.D., Farquhar, G., Foerster, J.N., Whiteson, S., 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. J. Mach. Learn. Res. 21, pp. 178:1–178:51.

Raza, S., Wang, S., Ahmed, M., Anwar, M.R., 2019. A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions. Wireless Communications and Mobile Computing 2019.

Raza, S., Liu, W., Ahmed, M., Anwar, M.R., Mirza, M.A., Sun, Q., Wang, S., 2020. An efficient task offloading scheme in vehicular edge computing. J. Cloud Computing 9 (1), 1–14.

Sheraz, M., Ahmed, M., Hou, X., Li, Y., Jin, D., Han, Z., Jiang, T., 2021. Artificial intelligence for wireless caching: Schemes, performance, and challenges. IEEE Commun. Surveys Tutorials 23 (1), 631–661.

Skondras, E., Michalas, A., Vergados, D.D., 2019. Mobility management on 5g vehicular cloud computing systems. Vehicular Communications 16, 15–44.

Spinelli, F., Mancuso, V., 2020. Towards enabled industrial verticals in 5g: a survey on mec-based approaches to provisioning and flexibility. IEEE Communications Surveys & Tutorials.

Sun, F., Cheng, N., Zhang, S., Zhou, H., Gui, L., Shen, X., 2018. "Reinforcement learning based computation migration for vehicular cloud computing". In: 2018 IEEE Global Communications Conference (GLOBECOM), Dec. 2018, pp. 1–6.

Sun, Y., Guo, X., Song, J., Zhou, S., Jiang, Z., Liu, X., Niu, Z., 2019. Adaptive learning-based task offloading for vehicular edge computing systems. IEEE Trans. Veh. Technol. 68 (4), 3061–3074.

Tokody, D., Albini, A., Ady, L., Rajnai, Z., Pongrácz, F., 2018. Safety and security through the design of autonomous intelligent vehicle systems and intelligent infrastructure in the smart city. Interdisciplinary Description of Complex Systems: INDECS 3-A, 384–396.

Ullah, S., Abbas, G., Abbas, Z.H., Waqas, M., Ahmed, M., 2020. Rbo-em: Reduced broadcast overhead scheme for emergency message dissemination in vanets. IEEE Access 8, pp. 175 205–175 219.

Wang, Z., Zhong, Z., Ni, M., 2018. "Application-aware offloading policy using smdp in vehicular fog computing systems". In: 2018 IEEE International Conference on Communications Workshops (ICC Workshops), May. 2018, pp. 1–6.

Xiao, Z., Dai, X., Jiang, H., Wang, D., Chen, H., Yang, L., Zeng, F., 2019. Vehicular task offloading via heat-aware mec cooperation using game-theoretic method. IEEE Internet Things J. 7 (3), 2038–2052.

Zhang, Y., Lopez, J., Wang, Z., 2019. Mobile edge computing for vehicular networks [from the guest editors]. IEEE Veh. Technol. Mag. 14 (1), 27–108.

Zhang, K., Leng, S., Peng, X., Pan, L., Maharjan, S., Zhang, Y., 2019. Artificial intelligence inspired transmission scheduling in cognitive vehicular communications and networks. IEEE Internet Things J. 6 (2), 1987–1997.

Zhao, J., Li, Q., Gong, Y., Zhang, K., 2019. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. IEEE Trans. Veh. Technol. 68 (8), 7944–7956.

Zhou, Z., Liu, P., Feng, J., Zhang, Y., Mumtaz, S., Rodriguez, J., 2019. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. IEEE Trans. Veh. Technol. 68 (4), 3113–3125.