**Frontiers in Computing and Intelligent Systems**
ISSN: 2832-6024 | Vol. 2, No. 3, 2022

# A Vehicle Service Migration Strategy Algorithm in 5G NR-V2X

**Lv Zhou***

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
* **Corresponding author**: Email: zhoulv2012@hotmail.com

**Abstract:** In recent years, a new technology called mobile edge computing (MEC) is proposed to alleviate mobile devices' increasing demands on computing resources and service latency. However, users may stay away from the fixed MEC server or even leave the service scope of the MEC server during the mobile process, which will lead to service performance degradation or service interruption. Such an interruption is intolerable for vehicle in V2X. Therefore, it is necessary to consider the issue of service migration. In this paper, we model the service migration problem in V2X as a multidimensional Markov decision process problem and consider a variety of metrics including computing cost, migration cost and energy consumption. A novel user mobility model is also considered to accurately reflect the motion state of users in reality. Then, a novel service migration algorithm is proposed to solve this multidimensional MDP problem. Our proposed algorithm applies Actor-Critic and entropy to ensure both convergence and exportability. Experimental results show that our proposed algorithm outperforms the baselines and has relatively strong robustness.

**Keywords:** Mobile edge computing; Vehicle edge computing; Service migration; Markov Decision Process; Delay; Energy consumption; Deep reinforcement learning.

## 1. Introduction

In recent years, with the rapid development of the Internet of Vehicles (IoV) and the continuous improvement of network infrastructure, Vehicle-to-Everything (V2X) applications (e.g., autonomous driving and real-time traffic monitoring, etc.) are gradually appearing in the public eye, providing many convenient aspects for driving. Meanwhile, a new technology called Mobile Edge Computing (MEC) has been proposed to meet these demands due to the increasing requirements of vehicles for computing resources and service latency. MEC servers are micro edge clouds deployed in mobile networks. Vehicles connect to roadside units (RSUs) equipped with MEC servers, and enjoy the convenience of low latency, high bandwidth, and sufficient computing resources. However, the vehicle may move away from the roadside unit or even out of the service range of the roadside unit during its driving movement, which will result in degraded service performance or service interruption. Especially in some services that are highly sensitive to latency, such as autonomous driving and road condition recognition, such service interruptions can affect the user experience. Thus, a model of service migration and an optimized solution need to be explored.

Service migration under V2X has been a topic in recent years and a lot of important work has been carried out by many researchers in this area. proposes a task offloading scheme to minimize the total energy consumption of mobile devices and edge servers, which the authors solve by a hybrid metaheuristic algorithm that yields a near-optimal solution. formulates the stochastic task offloading problem in a piecewise wireless access network as a Markov decision process that maximizes long-term utility performance by considering time-varying communication quality and computing resources. However, the above work cannot be directly applied to extremely dynamic vehicular networks. The Follow-Me Chain algorithm was proposed in the literature to solve the Service Function Chain (SFC) problem. In, the authors propose a reinforcement learning-based solution for offline RAN slicing and a low-complexity heuristic algorithm to meet the communication resource requirements of different slices, enabling maximum resource utilization.

Although the existing work has made great progress in terms of service migration strategies, there are still some issues that need to be further explored including several costs. All of these impact factors need to be taken into account to fully evaluate the effects of migration. More importantly, we should consider the long-term cumulative payback of migration to weigh the overall performance improvement. In order to solve this problem. Then, we model the migration process as a Markov decision process and consider following metric to measuring the cost of the migration process in this paper. In addition, this paper uses a mobility model to represent the movement state of the vehicle. Then, we propose a multi-agent deep reinforcement learning based service migration algorithm, which can effectively deal with solving the problem of excessive dimensionality in the traditional MDP problem. In addition, our proposed algorithm applies Actor-Critic network and entropy to ensure convergence and exportability. Finally, this paper conducts experiments and evaluates the algorithm based on real data sets.

## 2. System Model

In this paper, we consider a vehicle in a VEC network traveling in a one-way road equipped with $I$ roadside units with equal communication range, and each roadside unit is configured with a roadside unit $e$ of equal computing power. where $E = \{e_1, e_2, \dots, e_i\}$ denotes the set of all VECs set. We can divide the road section into $I$ segments, and each vehicle is randomly distributed in the urban area. We also consider the vertical distance between the road and the roadside unit as $d$. The ensemble of all vehicles can be denoted as $U =$

$\{u_1, u_2, \ldots, u_j\}$, where each vehicle $u \in U$ has a computing task, and the vehicles can choose to compute locally or offload to the roadside unit. These tasks are offloaded to the roadside unit via the roadside unit. Vehicles can be connected to a roadside unit $e \in E$ as they move within segment $i$ in Fig. 1.

Considering the tradeoff between cost and benefit during service migration, we introduce following relevant metrics to measure the cost: container migration cost, computing cost and energy consumption. Energy consumption is a key metric for electric vehicles. This is because it is one of the biggest concerns today. Usually the migration cost is expected to increase with the energy consumption.

In this paper, we also consider a scenario that vehicle in a VEC network traveling in a one-way road equipped with $I$ roadside units with equal communication range $r$, and each roadside unit is configured with a roadside unit $e$, where $E = \{e_1, e_2, \ldots, e_i\}$ Accordingly, we can divide the road section into $I$. We also consider the vertical distance between the road and the roadside unit as $d$. The set of all vehicles can be denoted as $U = \{u_1, u_2, \ldots, u_j\}$, where each vehicle $u \in U$ The ensemble of all vehicles can be denoted as, where each vehicle has a computing task, and the vehicles can choose to compute locally or offload to the roadside unit. Vehicles can be connected to a roadside unit $e \in E$ as they move within segment $i$. We show this VEC network in Fig. 1

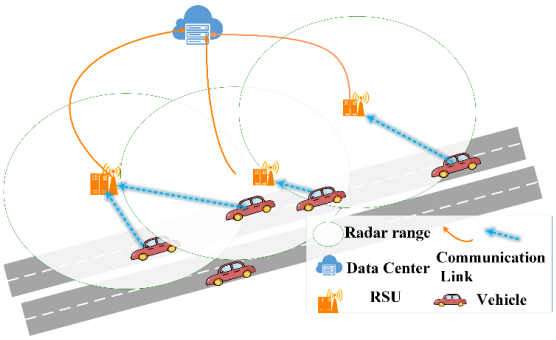



**Figure 1** V2X Network Architecture

## 2.1. V2X Communication Cost Model

In vehicle-to-infrastructure (V2I) communication, this paper considers vehicles communicating with roadside units via cellular networks and millimeter waves in V2X. The communication model depends on a certain distance between the vehicle and the roadside unit. According to the literature the distance between two network components is suggested to be 100 to 200 meters. Therefore, we assume a cellular link range of 200 meters. Also in this paper, we arrange the cellular network in mode 1 of NR-V2X . According to the 3GPP Rel.16 standard , NR V2X is capable of supporting advanced V2X applications with more stringent QoS requirements than those supported by cellular V2X. Therefore, in this paper, we classify the communication modes into 5G cellular and millimeter wave modes, which are described in detail in the following.

### 2.1.1. Millimeter wave mode

NR V2X uses millimeter wave (mm Wave) mode for NR. To take advantage of the benefits offered by the millimeter wave mode in V2I communication mode, in this paper we consider each vehicle to be equipped with a directional antenna array and employ directional beamforming to enhance the millimeter wave signal propagation. To maximize the directivity gain of the millimeter-wave antenna, we assume beam collimation for the transmitter and receiver and so we can approximate the directional antenna pattern as a sector model on an ideal horizontal plane .. In this paper we model the antenna gain as.

$$g_{u,e}(\eta) = \begin{cases} g_m, & |\eta| \le \frac{\eta'}{2} \\ g_s, & |\eta| > \frac{\eta'}{2} \end{cases} \quad (2)$$

where $\eta$ is the difference between the current antenna angle and the angle when the antenna gain is at its peak, i.e. the tolerable alignment error in the direction of antenna steering, $\eta'$ the beamwidth, the$g_{\square}$, and$g_{\square}$ are the directional gains of the main and side flap antennas, respectively.

Then we formulate the millimeter wave channel bandwidth as $C_{u,e}^{m} = B_m \log_2(1 + \mathrm{SINR}_{u,e})$, where$B_{\square}$ is the millimeter wave signal bandwidth; the SINR between the vehicle antenna and the RSU antenna is :

$$\mathrm{SINR}_{u,e} = p_{\mathrm{u}} - N_m - 10\log_{10}(B_m) + 2g_m - 10\alpha \log_{10}(d_{u,e}) - 69.6 - \rho) \quad (3)$$

where $p_{\mathrm{u}}$ is the transmission power of the millimeter-wave transceiver the vehicle $u$ equipped with. $N_{\square}$ is the noise power spectral density; the vehicle $u$ and the RSU $e$ Manhattan distance between the vehicle and the RSU is more appropriately reflected than the Euclidean distance . We combine the vehicle $u$ antenna and the RSU $e$ distance between the vehicle antenna and the RSU antenna $d_{u,e}$ modeled as the Manhattan distance. $\rho \sim N(0, \sigma^2)$ is the shadow fading model in decibels, and $\sigma$ is the standard deviation.

### 2.1.2. Communication Model

$$C_{u,e} = \lambda_c C_{u,e}^{c} + \lambda_m C_{u,e}^{m}$$
$$s.t. \lambda_c = \{0,1\}, \lambda_m = \{0,1\}, \lambda_c + \lambda_m = 1 \quad (4)$$

where$\lambda_{\square}, \lambda_m$ are the binary variables representing whether to use 5G cellular network or NR mode for communication, respectively? When $\lambda_{\square} = 1$ time, we consider that the vehicle$u$ uses 5G cellular network as the communication mode, so$\lambda_{\square} = 0$ and vice versa when$\lambda_{\square} = 0$ when$\lambda_{\square} = 1$ we consider that the vehicle $u$ uses NR mode as the communication method.

## 2.2. V2X Migration Cost Model

In this paper, we use the Platform as a Service (PaaS) paradigm and use Docker technology . This technology has mechanisms to enhance application portability and allows applications to be deployed everywhere without environmental differences. Then we can model the migration cost as following.

We consider that each vehicle under a V2X network contain has computing tasks and that the tasks are defined in the following binary.

$$T_{\square} = \{p_u, S_u\} \quad (5)$$

where $p_{\square}$ is the number of computing resources required to complete the task $T_{\square}$ the computing resources required to complete the task, and$S_{\square}$ represents the vehicle $u$ the size of the service image executed.

We assume that the vehicle $u$ unload service to a remote RSU $e$ with a ratio of $\omega_{u,e}^{\mathrm{vec}}$ and therefore $\omega_{\square}^{\mathrm{loc}} = 1 - \omega_{u,e}^{\mathrm{vec}}$ is the ratio that vehicle $u$ compute service locally. Considering the service image size, the service image size executed at the RSU $e$ is $S_{u,e}^{\mathrm{vec}} = \omega_{u}^{\mathrm{loc}} \times S_u$ . Thus, the migration cost can be calculated as:

$$G_{u,e}^{\text{mig}} = \frac{S_{u,e}^{\text{vec}}}{C_{u,e}} = \frac{\omega_u^{\text{loc}} \times S_u}{\lambda_c C_{u,e}^c + \lambda_m C_{u,e}^m}$$
$$s.t. \lambda_c = \{0,1\}, \lambda_m = \{0,1\}, \lambda_c + \lambda_m = 1 \quad (6)$$

## 2.3. V2X Computing Cost Model

In the following, we present two computing models, local computation and remote computation, to further specify our computing model.

1） Local computing

When a vehicle $u$ is on a local computing task, the computing cost depends on its available resources. We consider that $f_{\square}^{\text{loc}}$ is the computing resources of vehicle $u$, then the local computing time $t_{\square}^{\text{loc}}$ can be obtained as:

$$t_{\square}^{\text{loc}} = \frac{p_u}{f_u^{\text{loc}}} \quad (7)$$

1） Remote VEC computing

When local computing resources are strained or the computing load is too high, we plan to offload the service to a remote roadside unit for computation. Like many applications, i.e., road detection, smart braking, etc., the reception time of the computation results returned from the roadside unit is ignored because the size of the data transferred back from the roadside unit is negligible compared to the size of the output mirror. In this case, $t_{\square}^{\text{vec}}$ indicates that the service execution time can be expressed as:

$$t_u^{\text{vec}} = \frac{p_u}{f_u^{\text{vec}}} \quad (8)$$

where $f_u^{\text{vec}}$ denotes the computing resources the RSU allocated to the vehicle $u$, $p_{\square}$ is the number of computing resources required to complete the task $T_{\square}$.

We assume that the vehicle $u$ offloading service to a remote roadside unit$e$ with a offloading ratio of $\omega_{u,e}^{\text{vec}}$ and therefore $\omega_{\square}^{\text{loc}} = 1 - \omega_{u,e}^{\text{vec}}$ is the proportion of vehicle $u$ the proportion of services offloaded to be performed locally. Based on the service mirror size, we can derive the percentage of services executed at the remote RSU $e$ the mirror size of the services executed is $S_{u,e}^{\text{vec}} = \omega_u^{\text{loc}} \times S_u$. Thus, the cost is calculated as:

$$G_{u,e}^{\text{comp}} = \omega_u^{\text{loc}} t_u^{\text{loc}} + \omega_{u,e}^{\text{vec}} t_u^{\text{vec}} = \omega_{\square}^{\text{loc}} \frac{p_u}{f_u^{\text{loc}}} + \omega_{u,e}^{\text{vec}} \frac{p_u}{f_u^{\text{vec}}} \quad (9)$$

## 2.4. V2X Energy Consumption Model

2） Local computing

When a vehicle $u$ computes tasks locally, the computation time and energy consumption depend on its available resources. We consider $f_{\square}^{\text{loc}}$ as the available computing resources of vehicle $u$. Then, the local computation energy consumption $P_{\square}^{\text{loc}}$ can be calculated as following:

$$P_{\square}^{\text{loc}} = \vartheta_u \times p_u \quad (10)$$

where $\vartheta_{\square}$ denotes the energy consumption price of local computing resources.

3） Remote VEC computing

When the local computing resources are tight or the computing load is too high, we plan to offload the service to the RSU. In this case, the energy consumption of the RSU $e$ can be calculated as following:

$$P_{u,\text{e}}^{\text{vec}} = \vartheta_e \times \frac{S_{u,e}^{\text{vec}}}{C_{u,e}} \quad (11)$$

where $\vartheta_{\square}$ denotes the average transmission power of the vehicle $u$ during migration. $S_{u,e}^{\text{vec}}$ is the size of the service image, $C_{u,e}$ is the data transfer rate of vehicle$u$. Thus the vehicle $u$ energy consumption can be expressed as:

$$P_{u,\text{e}} = P_u^{\text{loc}} + P_{u,\text{e}}^{\text{vec}} \quad (12)$$

## 2.5. V2X Mobility Model

In this section, we consider the position of the vehicle relative to the RSU and the velocity of motion in this model. We assume that the vehicles are constantly moving and the corresponding RSU is always changing for better QoS. In this case, a vehicle mobility model is proposed to represent the state of vehicle movement state. As shown in Figure 3, the$M_{u,e}^t$ indicates the state of vehicle $u$ in time slot $t$.

$$M_{u,e}^t = \{x_{u,e}^t, y_{u,e}^t, v_u^t, \alpha_{u,e}^t\} \quad (13)$$

where $x_{u,e}^t$ and$y_{u,e}^t$ denote the distance between vehicle $u$ and the corresponding RSU $e$ on x-axis and y-axis in time slot $t$, $v_{\square}^t$ denotes the velocity of the vehicle $u$ which can be obtained from the sensors equipped on the vehicle, $\alpha_{u,e}^t$ represents the difference between the angle between the direction of travel of each vehicle on the x-axis and the angle between the line connecting the vehicle and the RSU on the x-axis.

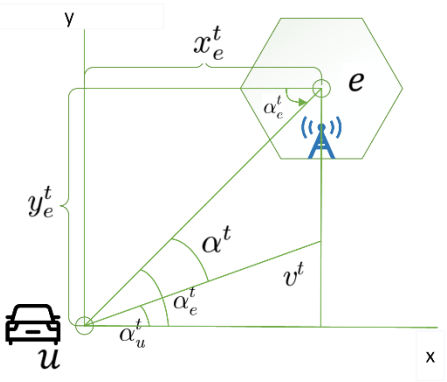



**Figure 2** Vehicle mobilty model

# 3. MARL-based 5G NR V2X service migration algorithm

## 3.1. Service migration problem transformation

Based on the above discussion, the service migration strategy should consider some intervention factors. Markov Decision Process (MDP) is a feasible model for developing a service migration process. In this model, an intelligence in a given state is informed of a set of possible actions it can take (e.g., migrate or not migrate), each of which is immediately rewarded or punished. Therefore, it must decide which action to take to maximize its reward, in other words, to minimize the penalty or cost in the long run.

In a multi-intelligent environment, the intelligences generally cannot fully observe the states of other intelligences, so multi-intelligent reinforcement learning is essentially considered as a partially observable Markov decision process (POMDP). In a POMDP with multiple intelligences, unless the strategies of other intelligences converge to, the environment of the intelligences is of a non-static nature, i.e., the intelligences cannot determine whether their own actions contribute to the learning outcome. To solve this problem, appropriate rewards must be given based on the information from other intelligences, otherwise, policy learning must be performed while keeping the multi-intelligent environment static.

#### 3.1.1. State Space

We use $S$ to represent the status of the vehicle in VEC network, defined as follows.

$$S \triangleq \{s \mid s = (T_u, M_{u,e}^t)\} \quad (14)$$

where $T_{\square}$ is the computing task contained in each vehicle and the task definition contains a binary set of the service image size and the computing resources required for the service, and$M_{u,e}^t$ is the mobility model of the vehicle.

#### 3.1.2. Reward Functions

In this paper, the intelligences will collect information about the environment and calculate the reward for each action, the specific formulation is as follows.

$$r_{u,e}^k = \omega_1(\Delta G_{u,e}^{\text{mig}} + \Delta G_{u,e}^{\text{comp}}) + \omega_2 P_{u,e}, \forall u \in U, \forall e \in E \quad (15)$$

where $G_{u,e}^{\text{mig}}, G_{u,e}^{\text{comp}}$ and$P_{u,e}$ are the estimated migration cost, computed cost and energy consumption, respectively. $\omega_i$ $(i = 1,2)$ represent the different weights that affect the service migration decision.

## 3.2. Algorithm Overview

Soft Actor-Critic (SAC) is an off-policy reinforcement learning algorithm that optimizes stochastic policies . This algorithm considers the maximization benefit between the expected benefit and entropy. This is closely related to the trade-off between exploration rate and utilization: increasing the value of this entropy leads to a higher exploration rate and also to a faster increase in utilization . In contrast to traditional deep Q-learning algorithms, it prevents the problem of fitting of strategies. Once an algorithm falls into this fitting trap, there is little that can be done to solve the overfitting problem.

Compared to previous deep reinforcement learning algorithms, the SAC algorithm provides effective sampling learning while retaining the benefits of entropy maximization and stability. In traditional reinforcement learning, the goal of the algorithm is to increase the value of the cumulative expected reward.

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \gamma^t [r(s_t, a_t)] \quad (16)$$

where$a_t$ represents the time gap in which the intelligent body$t$ the action taken by the$s_t$ represents the state of the intelligence in the time gap$t$ the state of the intelligence, and$\gamma$ denotes the reward value discount rate, and it can be concluded that the optimal strategy is

$$\pi_{std}^* = \arg\max_\pi \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} [r(s_t, a_t)] \quad (17)$$

Then, the entropy is introduced into Equation (22) with the goal of finding a most strategy to maximize this objective function. Thus, the optimal strategy can be expressed as follows

$$\pi^* = \arg\max_\pi \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \gamma^t [r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))] \quad (18)$$

where the temperature parameter$\alpha$ determines the relative importance of the entropy value to the reward value, the$\mathcal{H}(\pi(\cdot \mid s_t))$ represents the entropy, and the coefficient$\alpha$ can control the randomness of the optimal strategy.

## 3.3. Soft Q functions and soft values

The algorithm in this chapter consists of a Critic function and an Actor function. Where the Critic network is used to estimate the value function, containing the value of the action (Q-value) or the value of the state (V-value), while the Actor network updates the policy distribution in the direction suggested by Critic (e.g. with the policy gradient). According to soft Bellman equation, the long-term cumulative payoff can be expressed as:

$$V_{\text{soft}}(s_t) = \mathbb{E}_{a_t\sim\pi}[Q_{\text{soft}}(s_t, a_t) - \alpha\log(\pi(a_t \mid s_t))] \quad (19)$$

The corresponding Q function is:

$$Q_{\text{soft}}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1},a_{t+1}}[Q_{\text{soft}}(s_{t+1}, a_{t+1}) - \alpha\log(\pi(a_{t+1} \mid s_{t+1}))] \quad (20)$$

## 3.4. Policy Improvement

However, with the size of the problem grows, computing the value between functions becomes a challenge. Therefore, this chapter introduces the degree neural network (DNN) technique, which is widely used in the field of function approximation. In the strategy$\pi$ update process, this chapter uses a deep neural network to approximate the soft Q function $Q_{\text{soft}}(s_t, a_t)$ to further reduce the error value.$Q_\theta(s_t, a_t)$ denotes the soft approximated Q function, which is estimated by a parameterized DNN. This chapter also uses the replay buffer$\mathcal{D}$ to improve its performance.

$$\mathcal{D} \leftarrow \mathcal{D} \cup (s_{\square}, a_t, r(s_t, a_t), s_{t+1}) \quad (21)$$

Therefore, during the training process, the DNN can sample from the replay buffer to reduce the internal correlation between samples in order to avoid overfitting. Meanwhile, this algorithm combines the Q evaluation value$Q_\theta(s_t, a_t)$ and the Q target value$\hat{Q}(s_t, a_t)$ The average standard error between the Q-values is used as the performance index of DNN, while we create two Q-valued networks with$\theta_1$ and$\theta_2$ denoted. The loss functions of these two networks can be expressed as

$$J_Q(\theta) = \mathbb{E}_{(s_{\square},a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \quad (22)$$

where $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V_{\overline{\psi}}(s_{t+1})\right]$ , the $\mathcal{D}$ represents the replay buffer. The reason for introducing two state action value functions here in this chapter is to take the smallest one of them when updating the actors later to prevent overestimation, which is similar to the idea of Double-DQN . Then, this chapter computes the gradient of by deriving

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)(Q_\theta(s_t, a_t) - r(s_t, a_t) - \gamma V_{\overline{\psi}}(s_{t+1})) \quad (23)$$

The same soft value function is as follows.

$$J_V(\psi) = \mathbb{E}_{(s_{\square},a_t)\sim\mathcal{D}}[(V_\psi(s_t) - V_{\text{soft}}(s_t)]^2 \quad (24)$$

with

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(s_t)(V_\psi(s_t) - Q_\theta(s_t, a_t) + \log\pi_\phi(a_t|s_t)) \quad (25)$$

Under a set of action space collections for service migration, the improvement process for soft policies is as follows.

$$\pi' = \arg\min_{\pi_{\square}\in\Pi} D_{KL}\left(\pi_k(\cdot \mid s_t) \parallel \frac{\exp\left(\frac{1}{\alpha}Q_{soft}^{\pi}(s_t, \cdot)\right)}{Z_{soft}^{\pi}(s_t)}\right) \quad (26)$$

Where $\pi'$ denotes the new service migration policy, and $\pi$ denotes the old policy, and $\Pi$ denotes the total feasible set of policy functions, and the expectation value under state $s_{\square}$ under the state can be approximated by sampling from the Replay Buffer $\mathcal{D}$ to approximate the calculation, and $D_{\square\square}$ denotes the KL (Kullback-Leibler) scatter , and $Z_{soft}^{\pi}$ denotes the logarithmic partition function, which normalizes the distribution without affecting the policy gradient.

However, since the actions $a_{\square}$ are sampled from a random policy distribution, their gradients cannot be computed

directly. To solve this problem, this chapter introduces a reparameterization technique: $a_t = f_\phi(\varepsilon_t; s_t) = f_\phi^\mu(s_t) + \varepsilon_t \odot f_\phi^\sigma(s_t)$, where$\epsilon \sim \mathcal{N}(\epsilon)$ is a noise parameter.

The objective function can then be rewritten as follows.

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[ \log \pi_\phi \left( f_\phi(\epsilon_t; s_t) | s_t \right) - Q_\theta \left( s_t, f_\phi(\epsilon_t; s_t) \right) \right] \quad (27)$$

The policy gradient for service migration in this chapter can be calculated as follows.

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \, log \, \pi_\phi(a_t|s_t) + \left( \nabla_{a_t} \log \pi_\phi(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t) \right) \nabla_\phi f_\phi(\epsilon_t; s_t) \quad (28)$$

## 3.5. Algorithm Implement

**Table 1**. MARL-Based 5G NR V2X Service Migration Algorithm (MACSMA)

| | |
|---|---|
| 1: | **Input**: initial policy parameter $\phi$, Q function parameters $\theta_1, \theta_2$, V function parameters $\psi_1, \psi_2$ |
| 2: | **Output**: Model weights. |
| 3: | Initial replay buffer $\mathcal{D}$ |
| 4: | **for** each iteration **do** |
| 5: | **for** each environment step **do** |
| 6: | Observe state $T_{\text{⎚}}, M_{u,e}^t$ and select action $a_{\text{⎚}}$ from the policy to decide the direction of migration |
| 7: | Execute $a_{\text{⎚}}$ and observe next state $s_{t+1}$ |
| 8: | Calculate cost and and perform normalization. |
| 9: | Calculate reward $r(s_{\text{⎚}}, a_t)$ by Eq. (15) |
| 10: | Update replay buffer $\mathcal{D}$ by Eq. (21) |
| 11: | **end for** |
| 12: | **for** each gradient step **do** |
| 13: | Sample from replay buffer $\mathcal{D}$ |
| 14: | Calculate and update parameters by Eq. (22-25) |
| 15: | Update policy weights by Eq. (27) and Eq. (28) |
| 16: | Update target value network by $\overline{\psi} \leftarrow \tau\psi + (1-\tau)\overline{\psi}$ |
| 17: | **end for** |
| 18: | **end for** |

The whole experimental procedure is based on the python-gym environment. First, this paper designs a simulation environment and collects the state information in the environment. Second, this paper uses a Gym wrapper to transform the data collected in this paper into a specific data structure. Then, the intelligent body performs some actions according to the initial random policy and observes the state of the environment in the next time period. In this paper, an Actor network and four Critic networks are designed, consisting of a value network, a target value network and two Q-value networks, respectively. Based on the currently observed states$s_{\text{⎚}}$, the Actor network outputs the parameters of the probability distribution of all actions, and then the actions are based on the probability distribution $a_{\text{⎚}}$ Sampling is performed, and after the vehicle adopts the action$a_{\text{⎚}}$ After that, collect the next $s_{t+1}$ and $r_{t+1}$. The intelligence is then trained through multiple iterations such that its long-term cumulative reward is maximized. In this paper, $(s_{\text{⎚}}, a_t, s_{t+1}, r_{t+1})$ denotes experience and we puts this experience into replay buffer $\mathcal{D}$. The purpose of the replay buffer is to remove the correlation between experiences, since previous and future actions in reinforcement learning are usually strongly correlated. Then we update the Q-value network, V-value network and Actor, and repeats the above steps until the algorithm converges. Finally, the weights of the model are obtained, i.e., the optimal service migration strategy. The complete algorithm is shown in Algorithm 1.

# 4. Experimental results and discussion

In this section, the first subsection describes the experimental environment. Then the performance of the service migration algorithm proposed in this chapter is evaluated.

## 4.1. Experimental Environment

To evaluate the performance of the proposed algorithm in a real scenario, this chapter uses Alibaba cluster data rather than randomly generated data to simulate the load of MEC servers in a real scenario. In this dataset, there is resource usage of each machine, meta and event information of containers, and resource usage of each container. This chapter assumes that there is a cellular network with a number of RSUs. Each RSU is equipped with an MEC server and is deployed within a radius of 300 meters. Within the communication range, user devices can communicate with the RSU via a data link and obtain services from the MEC server.

In the simulation experiments, the maximum number of training sets for the proposed and baseline algorithms is set to 60,000 and the maximum number of attempts (episodes) is 25. The learning rate of the Adam optimizer is set to 0.01 and the discount factor γ is set to 0.95 in this paper.

## 4.2. Experiment Results

In this subsection, we first compare the average rewards of MACSMA with different vehicle number. The iterations of the algorithms are all set to 60,000 iterations. As can be seen in Figure 3, the convergence of MACSMA is achieved in the following vehicle density environments, and the algorithm performs better when the vehicle density is low. With the vehicle density increases, the convergence and performance of the algorithm decreases, but remains at a more stable level.

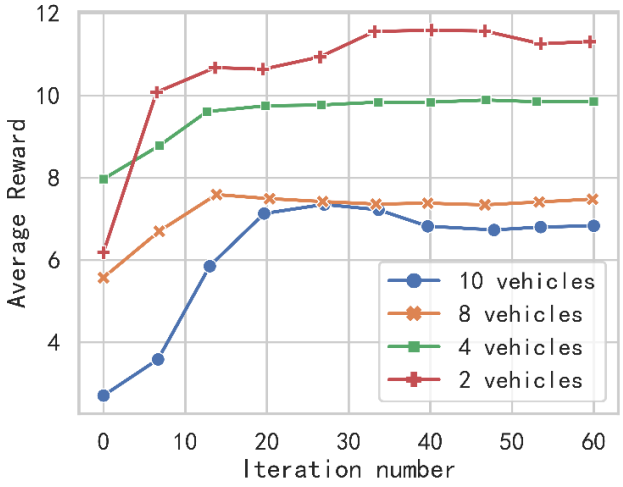



**Figure 3** Average reward with different vehicle number

Then, to further evaluate the performance of MACSMA, this section considers the following metrics.

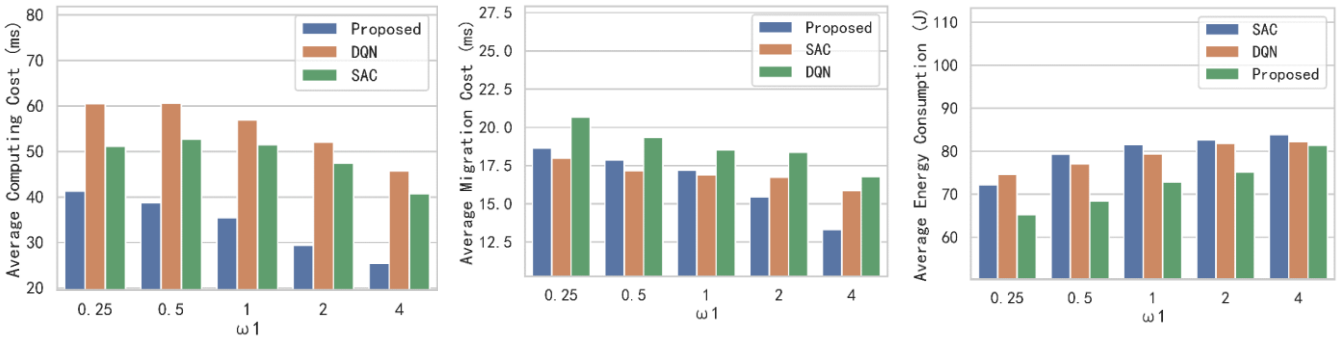
Migration Cost: The average time it takes to transfer a service from a transmitter to a receiver.

Calculated cost: the average time taken to perform the task.

Energy consumption: Average total energy consumption

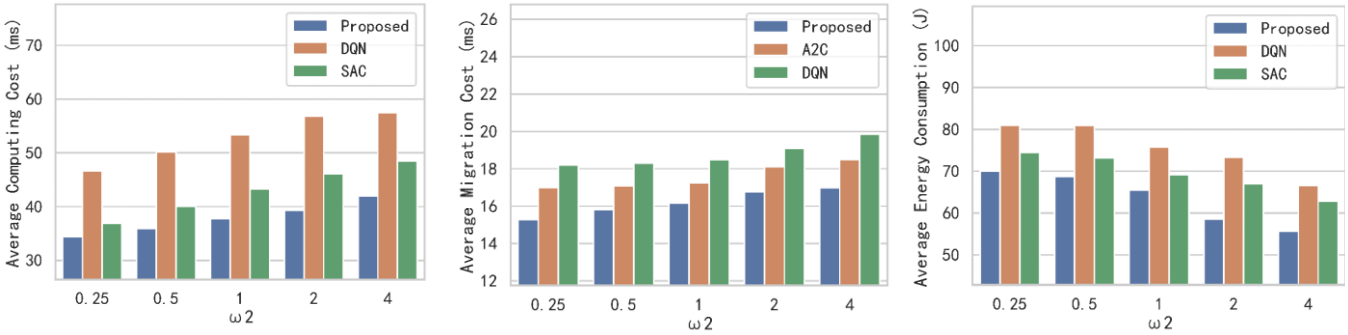
for service migration and service calculation.

This section also considers the sensitivity of different vehicles to delay and energy consumption, and by controlling $\omega_1$ and $\omega_2$ in Eq. (15) in to evaluate various types of performance.



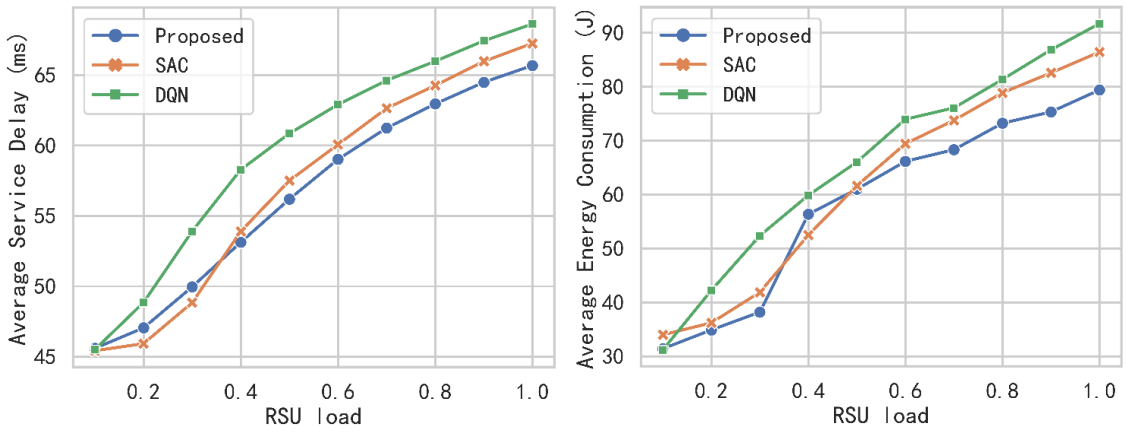

(a) (b) (c)

**Figure 4.** Average computing cost (a), migration cost (b), and energy consumption (c) with different $\omega_1$




(a) (b) (c)

**Figure 5.** Average computing cost (a), migration cost (b), and energy consumption (c) with different $\omega_2$




(a) (b)

**Figure 6.** Average service delay (a) and average energy consumption (b) with RSU load

Figures 4 and 5 illustrate the performance of service migration with different coefficient parameters. In Figure 4, when $\omega_2$ increases, the average computing cost of MACSMA decreases, because the larger $\omega_1$ is, the greater impact on service latency is, which is reasonable because $\omega_1$ affects the weight of service delay. However, this result can be easily deduced from Eq. (15) in this section. Therefore, the main purpose of this section to introduce the comparison into this paper is to investigate the performance of each algorithm under different service tendencies. This section fixes the performance of the algorithms by varying the$\omega_1$ the value of$\omega_2$ value of the algorithm, or fixing only the$\omega_2$ value while changing the$\omega_2$ values to change the weights of latency and energy consumption during service migration.

Figures 4 (a), (b) and (c) show the average migration cost, computing cost and energy consumption of the three algorithms. When $\omega_1$ is small, DQN performs the worst, while the algorithm in this paper performs the best in terms of average computing cost and energy consumption. However, as the$\omega_1$ increases, the performance of DQN improves, while the proposed algorithm in this paper is still the best in Fig. 4(b) and (c). In terms of energy consumption and migration cost, when $\omega_1$ is small, the algorithm in this paper clearly outperforms the other baselines. However, Fig. 6(c) shows that as $\omega_1$ increases, the advantage of this algorithm over other benchmark algorithms in terms of energy consumption is decreasing. The reason for this phenomenon can be attributed to the frequent migration problem, which is caused

by the requirement to increase the exploration rate to adapt to different environments. The pros and cons of this feature are discussed in this section below. In short, the benefits of a high exploration rate far outweigh its increased cost.

In Figure 5, this section focuses on investigating the effectiveness of the proposed algorithm with different energy consumption weights. This section finds that in Fig. 5(c), compared to other benchmark algorithms, the algorithm in this paper is very sensitive to energy consumption in terms of $\omega_2$ very sensitive. In other words, this section can be improved by changing the $\omega_2$ the energy consumption of the vehicle easily. In addition, it can be seen from Fig. 5(a) and (b) that the algorithm in this paper has an advantage in terms of migration delay and computing cost. Regardless of the $\omega_2$ of the values, our proposed algorithm in this paper and SAC are ranked second, which is because they are policy optimization methods and focus more on stability. Figures 5(a), (b) and (c) show that DQN performs relatively poorly due to the fact that DQN focuses on training the Q function, which is more sample efficient but also tends to lead to its poor stability.

Then, we investigate the average service delay and energy consumption under different RSU loads in Figure 6. In this paper, we set the RSU load to a value between 0 and 1. The higher the value is, the heavier the load of RSU is. From Figure 6 (a), we can see that the average service delay gradually increases as the load of the RSU increases. Since the available computing resources are limited, the RSU works at a higher load level, which means that the computing ability of the RSU decreases. The average service delay of our proposed algorithm in this paper is about 61ms at 0.8, which is 5ms less compared to the DQN algorithm. The reason is that the migration cost model in this paper facilitates the load balancing in resource pricing. In Figure 6 (b), the average energy consumption increases gradually with the increase of the RSU load. The reason is that multiple service migrations will be triggered to avoid waiting due to resource exhaustion. Overall, the performance of our proposed algorithm performs worse than SAC under low load, but its advantage increases with the increase of RSU load. The reason is that the single-agent algorithm has the disadvantage of not being able to consider the behavior of other vehicles, while our proposed algorithm considers the resource competition problem that exists in multi-vehicle scenarios and thus has better performance under high RSU load.

# 5. Conclusion

In this paper, we study the service migration problem under 5G-NR V2X, where vehicles migrate services through RSU and are optimized to improve computing efficiency by decreasing delay and energy consumption as much as possible. In this paper, we utilize the communication models of 5G NR-V2X, i.e., cellular link and millimeter wave, to improve the system performance by modeling the migration cost, computing cost, and energy consumption based on two communication models under V2X, and transforming the trade-off problem between the cost and benefit of service migration into a Markov decision process. Then this paper proposes a multi-intelligence learning algorithm based on reinforcement learning. Experiment shows that the algorithm proposed in this paper can improve both the service delay and energy consumption.

# References


[1] A. Kostopoulos, I. Chochliouros, J. Ferragut, Y. Ma, M. Kutila, A. Gavras, S. Horsmanheimo, K. Zhang, L. Ladid, and A. Dardamanis, "Use cases and standardisation activities for eMBB and V2X scenarios." pp. 1-7.

[2] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5G NR V2X communications," IEEE Communications Surveys & Tutorials, vol. 23, no. 3, pp. 1972-2026, 2021.

[3] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3774-3785, 2020.

[4] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4005-4018, 2018.

[5] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing, " IEEE Journal on Selected Areas in Communications, vol. 36, no. 10, pp. 2333-2345, 2018.

[6] H. D. R. Albonda, and J. Pérez-Romero, "An efficient RAN slicing strategy for a heterogeneous network with eMBB and V2X services, " IEEE access, vol. 7, pp. 44771-44782, 2019.

[7] K. Guan, G. Li, T. Kürner, A. F. Molisch, B. Peng, R. He, B. Hui, J. Kim, and Z. Zhong, "On millimeter wave and THz mobile radio channel for smart rail mobility," IEEE Transactions on Vehicular Technology, vol. 66, no. 7, pp. 5658-5674, 2016.

[8] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11 bd & 5G NR V2X: Evolution of radio access technologies for V2X communications, " IEEE access, vol. 7, pp. 70169-70184, 2019.

[9] M. Harounabadi, D. M. Soleymani, S. Bhadauria, M. Leyh, and E. Roth-Mandutz, "V2X in 3GPP standardization: NR sidelink in release-16 and beyond," IEEE Communications Standards Magazine, vol. 5, no. 1, pp. 12-21, 2021.

[10] B. Sklar, "Rayleigh fading channels in mobile digital communication systems .I. Characterization," IEEE Communications Magazine, vol. 35, no. 7, pp. 90-100, 1997.

[11] J. Wildman, P. H. J. Nardelli, M. Latva-aho, and S. Weber, "On the Joint Impact of Beamwidth and Orientation Error on Throughput in Directional Wireless Poisson Networks," IEEE Transactions on Wireless Communications, vol. 13, no. 12, pp. 7072-7085, 2014.

[12] Z. Li, L. Xiang, X. Ge, G. Mao, and H. C. Chao, "Latency and Reliability of mmWave Multi-Hop V2V Communications Under Relay Selections, " IEEE Transactions on Vehicular Technology, vol. 69, no. 9, pp. 9807-9821, 2020.

[13] Y. Oike, M. Ikeda, and K. Asada, "A high-speed and low-voltage associative co-processor with exact Hamming/Manhattan-distance estimation using word-parallel and hierarchical search architecture," IEEE Journal of Solid-State Circuits, vol. 39, no. 8, pp. 1383- 1387, 2004.

[14] L. Qing, "A 5G PaaS Collaborative Management and Control Platform Technology Based on Cloud Edge Collaboration Based on Particle Swarm Optimization Algorithm." pp. 144-147.

[15] A. Sergeev, E. Rezedinova, and A. Khakhina, "Docker Container Performance Comparison on Windows and Linux Operating Systems." pp. 1-4.

[16] R. Bellman, "A Markovian Decision Process," Journal of Mathematics and Mechanics, vol. 6, no. 5, pp. 679-684, 1957.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2018, pp. 1861--1870.

[18] C. Banerjee, Z. Chen, and N. Noman, "Improved Soft Actor-Critic: Mixing Prioritized Off-Policy Samples With On-Policy Experiences, " IEEE Transactions on Neural Networks and Learning Systems, 2022.

[19] R. S. Sutton, and A. G. Barto, Reinforcement learning: an introduction: MIT press, 2018.

[20] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning."

[21] S. Kullback, and R. A. Leibler, "On information and sufficiency," The annals of mathematical statistics, vol. 22, no. 1, pp. 79-86, 1951.