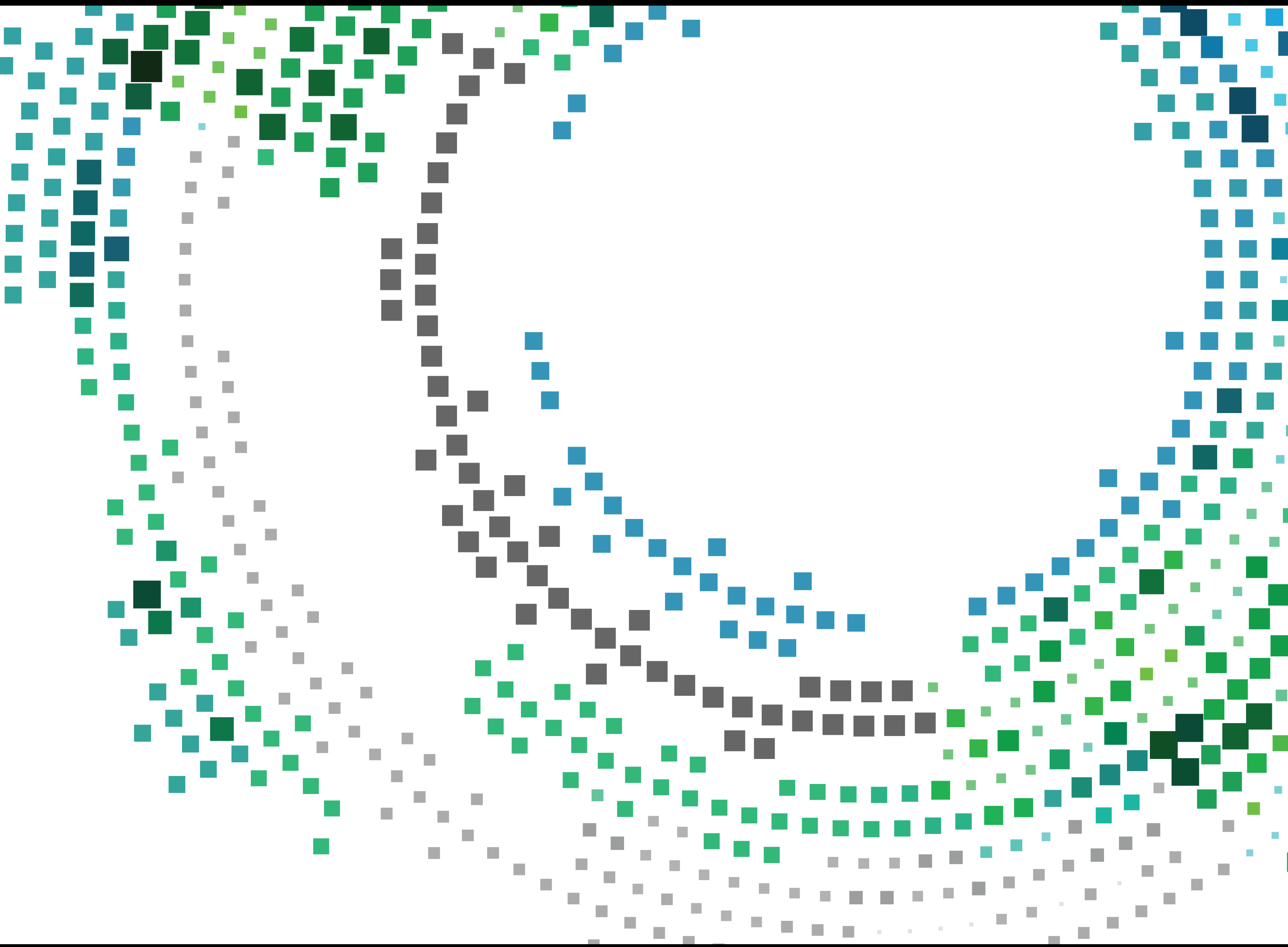
Mobile Information Systems

Multi-Agent Deep Reinforcement Learning for Unmanned Aerial Vehicles and the Internet of Vehicles

Lead Guest Editor: Konglin Zhu
Guest Editors: Yuming Ge and Lingxi Li



Multi-Agent Deep Reinforcement Learning for Unmanned Aerial Vehicles and the Internet of Vehicles

Multi-Agent Deep Reinforcement Learning for Unmanned Aerial Vehicles and the Internet of Vehicles

Lead Guest Editor: Konglin Zhu
Guest Editors: Yuming Ge and Lingxi Li

Contents

Hindawi
Mobile Information Systems
Volume 2022, Article ID 4988266, 13 pages
https://doi.org/10.1155/2022/4988266

Research Article

DIMDP: A Driving Intention-Based MDP Service Migration Model under MEC/MSCN Architecture

Lei Ye,[1] Kaiwen Ling,[1] Qingwen Han,[1] Yufei Yan,[1] Lingqiu Zeng,[2] Lingfeng Qi,[1] Li Lin,[1] and Junjun Zhang[2]

[1]*School of Microelectronics and Communication Engineering, Chongqing University, Chongqing 400044, China*
[2]*College of Computer Science, Chongqing University, Chongqing 400044, China*

Correspondence should be addressed to Qingwen Han; hqw@cqu.edu.cn

Received 15 July 2022; Accepted 26 September 2022; Published 11 October 2022

Academic Editor: Konglin Zhu

Service migration is one of the key topics of Internet of Vehicles (IoV). MEC (mobile edge computing), which is carried on the roadside unit (RSU), could serve as a service provider and provide a V2I (vehicle to infrastructure) cooperation service. To solve the high migration rate caused by the vehicle's high mobility feature, MSCN (mobile secondary computing node) framework is defined. To study service migration features under the MSCN framework further, in this paper, road vehicle's motion features, e.g., driving intention and regional traffic condition, are introduced to construct the Markov decision model, which is used to explain the service migration decision procedure, while DIMDP (driving intention-based MDP) is defined. Corresponding cost functions are defined, while the optimal object is given. A two-way road scenario is selected as a typical scenario. NS3 platform is employed to fulfill the simulation process. Simulation results show that the proposed service migration strategy performs well and is intensive to vehicle density change.

1. Introduction

With the rapid growth of vehicles, road safety has become a problem faced by countries all over the world. Vehicle safety applications by means of real-time information exchange between vehicles are techniques for avoiding traffic accidents.

Although cooperative vehicle infrastructure system (CVIS) is considered an effective approach to support vehicle safety applications, it has to meet the requirements of real-time ability, reliability, and service continuity.

For non-local service requests, the traditional centralized cloud computing network has defects in terms of delay and throughput. Therefore, in order to match the needs of real-time driving decision making, scholars have introduced edge computing technologies such as mobile edge computing (MEC) and fog computing (FC) into CVIS. Also, the edge computing server is configured on the side close to the user, deployed at the end of the base station (BS) and roadside unit (RSU). The back-end service information can be calculated and maintained at the edge node, thus providing relatively adequate computing ability [1] and supporting computational-intensive and time-sensitive operations or flexible deployment of applications and services. But for non-local services, MEC needs to maintain a certain service migration rate to reduce service delay. However, the high mobility of vehicle nodes is a negative impact on the efficiency of feedback information and service migration rate. The reason is that the object vehicle may leave the coverage area of the previous MEC, which undertakes computing tasks. To overcome this problem, in our previous work [2], a new concept, mobile secondary computing nodes (MSCNs), was defined, while a three-layer framework is constructed.

Obviously, the relative speed of MSCN and its surrounding vehicles is lower than that of RSU and its surrounding vehicles and partially solves the service migration problem. However, the corresponding migration mechanism is different from that of traditional MEC architecture. Hence, we discussed the service migration problem under MSCN architecture [3] and proposed a MDP (Markov decision process)-based service migration strategy.

However, in [3], the service migration strategy was designed according to the relative distance between vehicles and did not consider the vehicle's moving feature and motion model. Obviously, service migration decision making is influenced by the traveling trace of the vehicle, which is the reflection of driving intention and is affected by regional vehicle density. Since the vehicle prefers to go to places with good traffic conditions, we can estimate the vehicle's intention by considering the vehicle's moving feature and motion model.

Furthermore, the MDP model is considered an effective approach to express service migration progress. By establishing a state space that conforms to the vehicle's moving feature and combining the motion model, a trade-off is made between the cost of migration and the benefits after migration to obtain the optimal service migration strategy.

Therefore, in this paper, driving intention-based MDP service migration strategy under MSCN architecture is proposed. Specifically, our research contributions are as follows:

According to vehicle driving intention and regional traffic condition factors over the dynamic feature of service requirement, a MDP-based service migration method, DIMDP (driving intention-based MDP), is proposed.

A two-way road scenario, which includes four microscopic events, namely, following, turn left, turn right, and U-turn, is selected as the typical scenario.

The rest of the paper is structured as follows. Section 2 presents the related work, and Section 3 discusses the service migration using MDP based on MSCN. The service migration algorithm under MSCN is proposed and applied to traffic for service migration as an example in Section 4. Then, experimental simulation and result analysis are presented in Section 5. Finally, the conclusion is given in Section 6.

2. Related Work

We briefly investigated the existing literature on service migration from the perspective of service delay and trajectory prediction.

Yu et al. [4] prioritized MEC services and proposed a partial dynamic optimization algorithm (PDOA) service migration strategy calculation algorithm, which calculates the migration strategy that minimizes the average delay of long-term service by predicting the mobility of vehicles and achieves good results when the density of vehicles is small. However, the algorithm does not consider the impact of service interruption caused by the long migration time when the vehicle density is high.

Ge et al. [5] proposed the FEE algorithm to calculate the best service migration strategy, which takes into account the current and future time delays in N time slots and the service migration interface between vehicles and changes the focus of the migration strategy by setting up weight coefficients for the delay of each time slot. The algorithm can achieve better results by increasing the number of time slots, which is not obvious when the number of time slots is 5. However, the algorithm does not consider the frequency of service requests, so the algorithm may not achieve better results when services require frequent requests.

Nadembega et al. [6] proposed a mobility-based service migration prediction (MSMP) model, which split user requested service into several portions for service migration by estimating the throughput that the user could receive in advance.

Xu et al. [7] investigated path selection for seamless service migration and proposed a path-selection algorithm to jointly optimize both interests of the network plane and service plane and designed a distance-based filter strategy to eliminate undesired switches in advance to improve the scalability of the proposed algorithm.

Recently, the MDP model is considered an effective approach to express service migration progress. Corresponding researchers considered both user mobility features and MEC-based service requirements to establish decision-making strategies [8]. Moreover, cost function definition methods are proposed [9]. Combined with MDP and cost function definition. Taleb et al. [10] proposed a service migration decision algorithm to verify the effectiveness of Follow Me Cloud and provided guidelines for the method in this paper.

However, the existing MDP-based research has not built a model suitable for the CVIS, which considers the vehicle's moving feature and motion model.

In view of traffic simulation, vehicle motion models are categorized into microscopic [11] and macroscopic [12]. Microscopic models focus on the dynamic features of the separate vehicles, while macroscopic models consider traffic features, such as the traffic density, average velocity, flow, and so on. Hence, in this paper, we introduce density factor into the vehicle intention prediction model, thus constructing a microscopic motion model. On the other hand, to establish the MDP model, we borrow from the idea of the cellular model [13], which has adopted the time-space discretion method, to construct a microscopic motion model.

3. Service Migration under MSCN Framework

According to the percentage of migrated content, service migration can be divided into partial migration and full migration. Different migration methods bring different advantages and disadvantages. Since partial migration involves real-time decisions about which part of the data to migrate, this paper only considers the overall migration of the service.

3.1. MSCN Framework.

The MEC-based architecture, MSCN, proposed in our previous work is shown in Figure 1.

Here MSCN framework includes three layers, which are MEC layer, MSCN layer, and general vehicle layer. Experimental verification shows that MSCN can provide reliable RSU-oriented services and significantly improves both communication performance and computing efficiency [2].

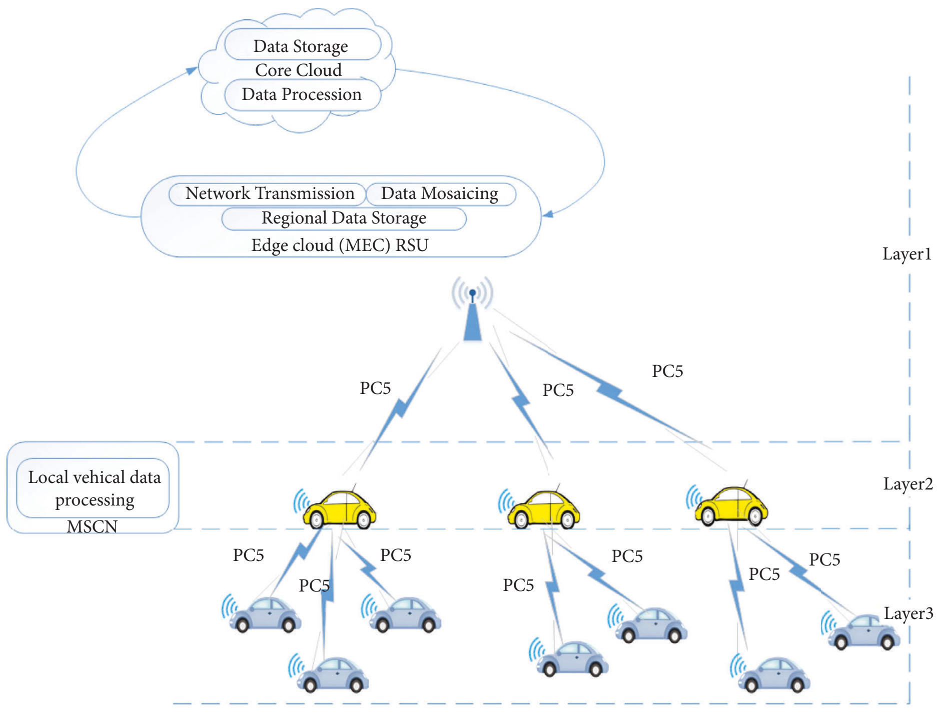


Figure 1: MSCN framework.

3.1.1. MEC Layer. The MEC layer provides computing and storage resource to road vehicle. Meanwhile, the service provider could provide vehicle-oriented service via the MEC interface.

3.1.2. MSCN Layer. Selected vehicle nodes, such as city buses, taxis, and so on, are used to construct the MSCN layer, which could provide local computing resources. MSCN nodes collect BSMs sent by surrounding vehicles, then fulfill information fusion, and upload corresponding results to the MEC layer.

3.1.3. General Vehicle Layer. The general vehicle layer includes all road vehicle nodes, which are equipped with DSRC/LTE-V blocks, and could communicate with each other and RSU directly.

3.2. MDP Model Based on Two-Way Lane. In this paper, we select a two-way lane scenario to discuss the service migration problem.

The problem involves the following assumption conditions:

Object region is full C-V2X coverage, which means that all on-road vehicles could communicate with RSU.

All road vehicles are equipped with a C-V2X module and communicate with RSU via the PC5 interface.

There are enough public traffic vehicles to serve as MSCNs, which communicate with RSU and other on-road vehicles via C-V2X technologies.

RSUs communicate with each other via wired networks, such as fiber backbone and Ethernet.

The relationship between users and service providers is expressed by the distance factor. A service migration event should occur when $d(t) > N$. In this paper, the maximum value of N is set as 10.

MSCN's dwelling time is subject to an exponential distribution with a mean value equal to $1/\mu$.

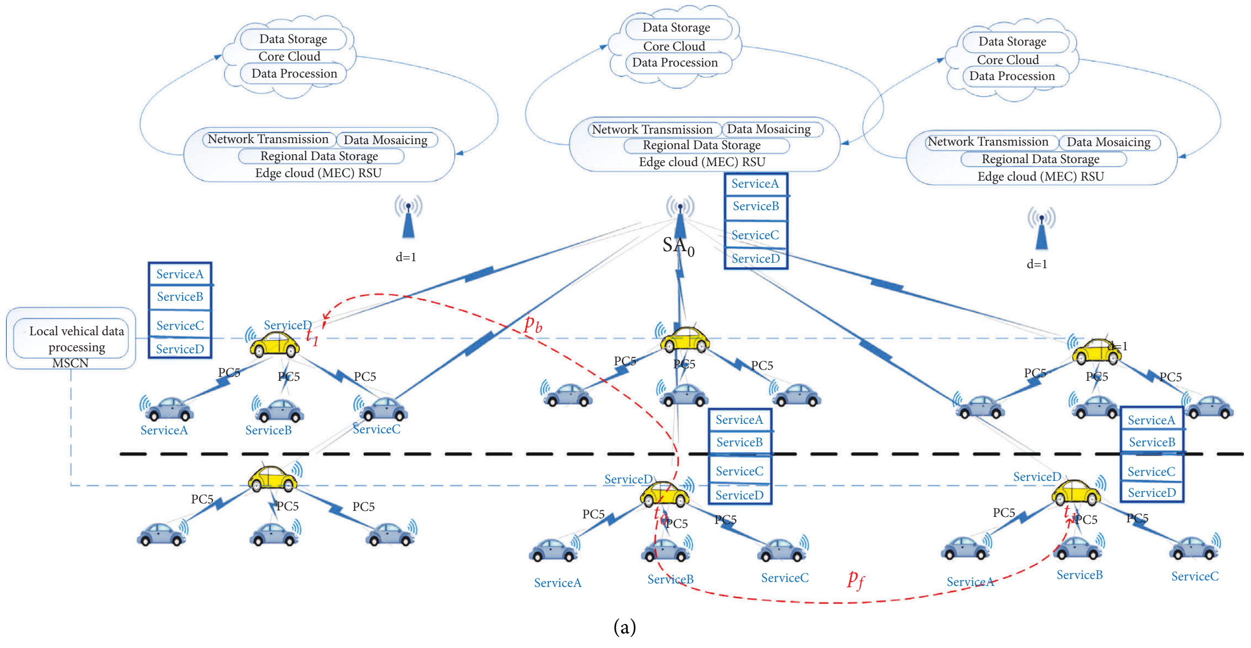
According to the above assumptions, the two-way lane MDP model is constructed. State space of service migration under the MSCN framework is shown in Figure 2(a), while corresponding state transition progress is denoted in Figure 2(b). As shown in Figure 3, service migration progress under MSCN architecture includes two migration levels, is denoted as social vehicle-oriented migration and public vehicle-oriented migration.

Social vehicle-oriented migration: a social vehicle changes its association MSCN (public vehicles), for example, MSCN1⟶MSCN2 as shown in Figure 3(a). It should be noted that social vehicle can only migrate to the MSCN with the same cruising direction.

Public vehicle-oriented migration: a public vehicle (MSCN) changes its association RSU, for example, RSU1⟶RSU2 as shown in Figure 3(b).

As shown in Figure 2(b), for an initial state $\pi_n, n \in [0, 4N-1]$, two inclining migration events are expressed as

$$\begin{cases} \pi_{n+1}, n \in [0, 2N-1], & \text{cruising forward}, \\ \pi_{n+1}, n \in [2N, 4N-1], & \text{cruising backward}. \end{cases} \quad (1)$$



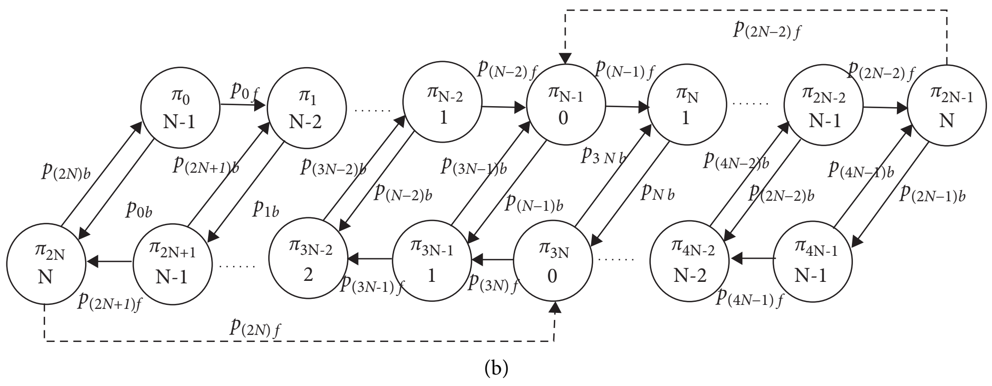


Figure 2: MDP model based on two-way lane. (a) State space of service migration. (b) State transition progress.

State transition probability matrix is defined as

$$P = \begin{pmatrix} p_{01} & \cdots & p_{0(4N-1)} \\ \vdots & \ddots & \vdots \\ p_{(4N-1)0} & \cdots & p_{(4N-1)(4N-1)} \end{pmatrix}, \quad (2)$$

where $p_{ij}, i, j \in [0, 4N-1]$ is the transition probability of state i and j. In this paper, we take into account the driving intention of the vehicle to obtain the transition probability in the next section.

Assume that state probability of state π_n is χ_n, and

$$\sum_{n=0}^{4N-1} \chi_n = 1. \quad (3)$$

State transition progress could be denoted as

$$\chi_f = \chi_f \cdot P. \quad (4)$$

Thus, we can obtain the row probability vector χ_n of states at the migration strategy N according to equations (3) and (4).

3.3. Vehicle Motion Features. Obviously, service migration progress is decided by vehicle motion features, e.g., cruising direction, cruising speed, and relative speed. For MSCN architecture, two kinds of relative speed, the relative speed between social vehicles (users) and public vehicles (MSCN) and relative speed between public vehicles (MSCN) and RSU, should be considered.

According to [14], for traffic participants, the probability of the specific event in the time interval $[t, t+\Delta t]$, is defined as

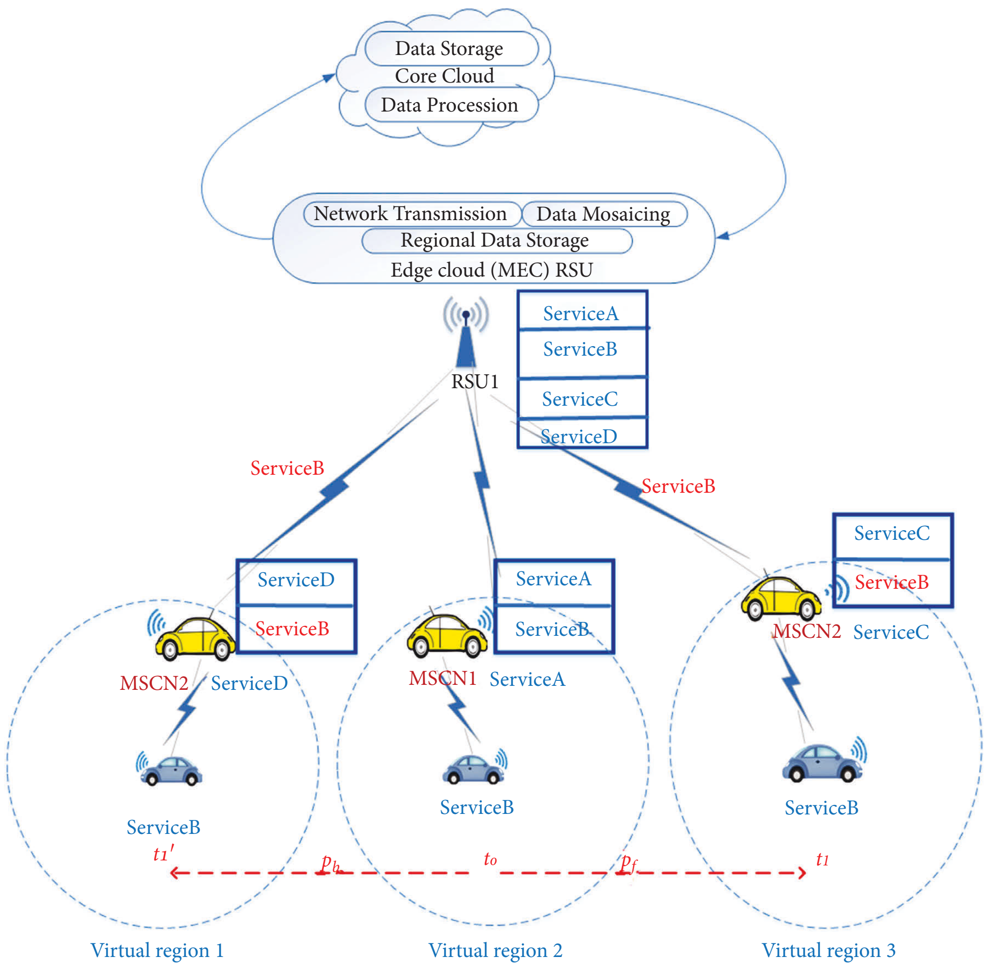
$$P(t; \Delta t) = \tau^{-1}[I(t)]\Delta t, \quad (5)$$

where $\tau^{-1} \in [0, \tau_{\max}^{-1}]$ is the number of events that occur per unit time interval under a certain condition and I is the situation-related risk indicator function.

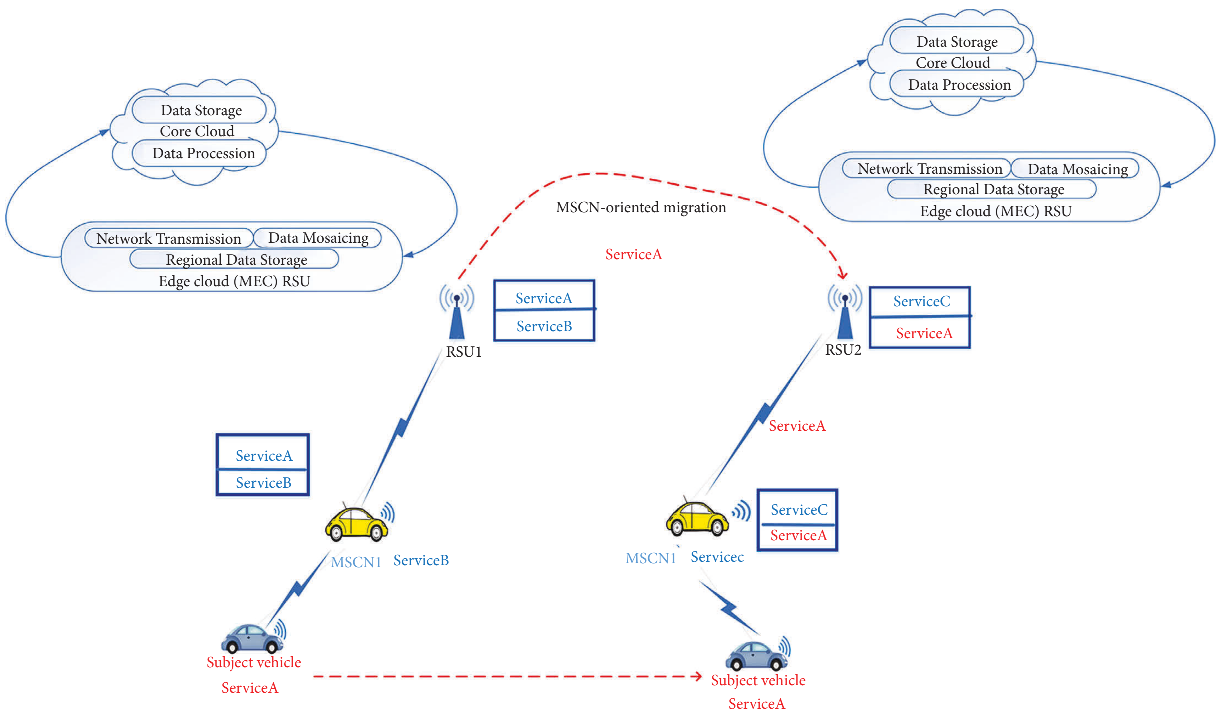
Although equation (5) is constructed for driving risky evaluation, it could be used to express state transition probability. Hence, we define a situation indicator function I_s, assuming that traffic prefers to go to places with better traffic conditions and regional traffic condition C_{traffic}, to replace risk indicator function I, as shown as follows:

$$I_s \sim (d_I, C_{\text{traffic}}), \quad (6)$$

where d_I is obtained according to path planning information and $C_{\text{traffic}} \in (0, 1]$ is real-time traffic information, defined as



(a)



(b)

Figure 3: Two levels of service migration under MSCN. (a) Social vehicle-oriented migration. (b) Public vehicle-oriented migration.

$$C_{\text{traffic}(t)} = \begin{cases} \max\left(e^{\frac{-v_{\Re(t)}(\bar{v}_{\text{MSCN}},0)}{v_{\text{limit}}}}, \frac{v_{\Re(t)}(\bar{v}, v_{\text{MSCN}})}{v_{\Re(t)}(\bar{v}_{\text{MSCN}},0)} \right), & v_{\Re}(\bar{v}_{\text{MSCN}},0) \neq 0, \\ 1, & \text{else}, \end{cases} \quad (7)$$

where $v_{\text{re}(t)}(\bar{v}, \bar{v}_{\text{MSCN}})$ is the relative speed between social vehicles (users) for time t and public vehicles (MSCN), while $v_{\text{re}(t)}(\bar{v}_{\text{MSCN}}, 0)$ is the relative speed between public vehicles (MSCN) and RSUs for time $t.v_{\text{limit}}$ is the maximum speed limit on road. $\bar{v}$ is the average cruising speed of vehicles in coverage of corresponding MSCN. $\bar{v}_{\text{MSCN}}$ is the average cruising speed of MSCN in coverage of corresponding RSU.

The worse the traffic condition is, the higher the C_{traffic} value should be. Note here that we assume that path change is a result of bad traffic conditions.

$$d_I(t + \Delta t) = d_I(t) * \frac{\exp\left[1 - C_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right]}{\exp\ \left(1 - C_{\text{traffic}}(t)\right)}. \quad (8)$$

In view of service migration, according to vehicle cruising direction, the migration area could be divided into two parts, namely, the forward area and the backward area. As mentioned earlier, here we consider four kinds of events: the following, turn left, turn right, and U-turn. Here we categorize the following, turn left, and turn right as forward area migration inclining events while U-turn as backward area migration inclining events.

Then, we define vehicle states as $\pi_n, n \in [0, M-1]$, where M is the number of states, and $p_{ij}, i, j \in [0, M-1]$, is the state transition probability, which is denoted as

$$p_{ij} = P\left(\pi^{\Delta t} = \pi_j \mid \pi = \pi_i\right) \sim \tau^{-1}\left(d_I, C_{\text{traffic}}\right)\Delta t, \quad (9)$$

where π_i is the state of time t and π_j is that of time $(t + \Delta t)$.

Considering vehicle's inclining events, forward or backward, and with the assumption that traffic prefers to go to places with better traffic conditions and regional traffic conditions C_{traffic}, the state transition probability proportional to d_I is defined as follows:

$$\begin{cases} \frac{p_{if}}{d_I^f} = \frac{p_{ib}}{d_I^b}, \\ p_{if} + p_{ib} = 1, \end{cases} \quad (10)$$

where p_{if} and p_{ib} are occurrence probabilities of forward event state π_f and forward event state π_b.

Then, p_{if} and p_{ib} could be denoted as

$$\begin{cases} p_{if} = \frac{d_I^f}{d_I^f + d_I^f}, \\ p_{ib} = \frac{d_I^b}{d_I^f + d_I^f}. \end{cases} \quad (11)$$

Substitute (8) into (11); then,

$$\begin{cases} p_{if} = \frac{\exp\left[1 - C^f_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right]}{\exp\left[1 - C^b_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right] + \exp\left[1 - C^f_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right]}, \\ p_{ib} = \frac{\exp\left[1 - C^b_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right]}{\exp\left[1 - C^b_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right] + \exp\left[1 - C^f_{\text{traffic}}(t + \Delta t)/C_{\text{traffic}}(t)\right]}. \end{cases} \quad (12)$$

4. Service Migration Algorithm and Typical Applications under DIMDP-Based MSCN Architecture

4.1. Migration and Transmission Cost. Obviously, the optimization target of the service migration strategy is to minimize migration and transmission costs, and thus it could improve service performance.

Here we assume that the dwelling time of vehicle node i is calculated as

$$T_{\text{dewll,vehicle}}(i) = \frac{r_{t,rsu}}{v_{\text{vehicle}}(i)}, \quad (13)$$

where $r_{t,rsu}$ is the coverage radius of vehicle i's home RSU and $v_{\text{vehicle}}(i)$ is the cruising speed of vehicle i.

We define two cost functions, namely, transmission cost function $C_{\text{trans}}(d)$ and migration cost function $C_m(N)$, as follows:

$$\begin{cases} C_{\text{trans}}(d) = \dfrac{D_{trans}(d) \cdot r_{t,rsu} \cdot f}{v_i}, \\ C_m(N) = D_m(N) + I_m(N), \end{cases} \tag{14}$$

where $D_{\text{trans}}(d)$ is the message transmission delay between the user vehicle node and service provider, at a distance d, (f) is the packet transmission frequency, and N is the migration distance. Migration progress should occur if $d > N$. $D_m(N)$ is the migration delay function, while $I_m(N)$ is the service interruption function, defined as follows:

$$\begin{cases} D_m(N) = \dfrac{\theta_m}{R} + \dfrac{(\beta+1)\lambda_f}{R}(N+1), \\ I_m(N) = \exp\left(\dfrac{D_m(N)}{k}\right) - 1, \end{cases} \tag{15}$$

where R is the data transfer rate, λ_m is the size of the data that needs to be transmitted during the service request process, λ_f represents the size of data frame structure waiting to be sent in the queue, and β represents the number of the frame structure, which is affected by network congestion.

Message transmission delay is the sum of dissemination delay $D_t(d)$ as shown in (16) and queuing delay $D_q(d)$, as shown as

$$\begin{cases} D_t(d) = \dfrac{\lambda_m}{R} + d_c, \\ D_q(d) = \dfrac{(\beta+1)\lambda_f}{R} d, \end{cases} \tag{16}$$

where d_c, which is a constant, represents the wireless communication delay between the vehicle and the RSU.

Then, we define $D_{\text{trans}}(d)$ as

$$D_{\text{trans}}(d) = D_t(d) + D_q(d) = \frac{\lambda_m}{R} + \frac{(\beta+1)\lambda_f}{R} d + d_c. \tag{17}$$

Average total cost function, where the state probability of state π_{n}, χ_n, is used as a weight to calculate the weighted average of cost, is defined as

$$\begin{aligned} \overline{C_a}(N) = {} & \sum_{d=1}^{N-1} C_{\text{trans}}(d)\left(\chi_{N-1\pm d} + \chi_{3N-1\pm d}\right) \\ & + C_{\text{trans}}(N)\left(\chi_{2N} + \chi_{2N-1}\right) \\ & + C_m(N)\left(\chi_{2N} p_{(2N)f} + \chi_{2N-1} p_{(2N-1)f}\right), \end{aligned} \tag{18}$$

while the optimal objective is to minimize the average total cost as follows:

$$P1\colon S = \arg\min_N \overline{C_a}(n). \tag{19}$$

The pseudocode of optimal strategy is shown in Algorithm 1, while corresponding variables and functions are explained in Table 1. Firstly, the algorithm obtained the state transition probability matrix P according to vehicle driving intention and obtained the row probability vector χ_n according to (3) and (4) in different migration distance N. Then, the average total cost $\overline{C_a}$ is obtained according to (18). Finally, we compare the total cost at each migration distance to obtain the optimal service migration distance corresponding to the minimum average total cost $\overline{C_a}$.

Note here that to avoid the impact of the sudden speed change on service migration performance, in this paper, we use regional average speed as the input value of vehicle speed.

4.2. Service Migration Strategy

4.2. Service Migration Strategy. The proposed MDP-based service migration strategy under the MSCN framework is shown in Algorithm 2, and the corresponding variables and functions are explained in Table 2. First, the current position of each vehicle is obtained and compared with the position of the previous moment to determine whether the MSCN directly under the vehicle has changed. After that, if the MSCN directly under the vehicle changes, determine whether the RSU directly under the vehicle has changed. Then, if the RSU directly under the vehicle changes, compare the distance from the current distance with the optimal strategy obtained by Algorithm 1. Finally, once the distance to the optimal policy is reached, service migration is performed and P is updated, with the new service provider as the center. This completes a service migration.

According to the 3GPP suggestion, corresponding parameters are set as

$$\begin{cases} \lambda_m = 1200\text{bit}, \\ \theta_m = 3.5\text{Mb}, \\ \lambda_f = 1024\text{bit}, \\ \text{R} = 5\text{Mbps}, \\ f = 10Hz. \end{cases} \tag{20}$$

The relationship between migration distance N and migration cost with different p values is shown in Figure 4. Here regional average speed is set as 20 m/s.

As shown in Figure 4, total cost increases with increasing p, and thus we verify the validity of the cost function. Moreover, the optimal object S, which is defined by (19), varies from 3 to 4.

The relationship between migration distance N and migration cost with different regional average speed values is shown in Figure 5.

As shown in Figure 5, the average total cost increases with increasing regional average speed, which expresses the influence of the vehicle's cruising intention. The value range of optimal object S is [2, 5] and consistent with the result of Figure 5.

4.3. Intersection Control under the MSCN Framework. Intersection control in the MSCN framework leverages RSU collaboration across multiple intersections and decides

Table 1: Explanation for Algorithm 1's variables and functions.

Variable/function	Explanation
pif	Probability set of forward area migration inclining events
λ_m	Service request data size
θ_m	Service migration data size
λ_f	The data size of frame structure
$r_{t,rsu}$	The coverage radius of RSU
v_{vehicle}	Vehicle speed
f	Service request frequency
$C_{\min}$	The minimal average total cost $\overline{C_a}$
S	The optimal migration strategy, a service migration event occurring when $d(t) \geq S$
get_Pmatrix (pif, N)	Obtain the state transition probability matrix P according to pif at migration distance N
get_state (P, N)	Obtain the row probability vector χ_n of states at the migration distance N according to (3) and (-4)
Get_cost (Cset, χ_n, N)	Obtain the average total cost $\overline{C_a}(N)$ according to (18). Cset is $\{\lambda_m, \theta_m, \lambda_f, r_{t,rsu}, v_{\text{vehicle}}, f\}$.

```
Input:
    $p_{if}$, $\lambda_m$, $\theta_m$, $\lambda_f$, $(r)$, $r_{t,rsu}$, $v_{\text{vehicle}}$, $(f)$
Output:
    Optimal migration strategy $S$
$C_{\min} \leftarrow \infty$;
$S \leftarrow 10$;
for $N = 1$ to 10 do
$P \leftarrow$ get_Pmatrix ($p_{if}$, $N$);
$\underline{\chi_n} \leftarrow$ get_state ($P$, $N$);
$\overline{C_a} \leftarrow =$ get_cost (Cset, $\chi_n$,N);
if $C_{\min} > \overline{C_a}$
    $C_{\min} \leftarrow \overline{C_a}$;
    $S \leftarrow N$;
else
    break;
end
end
Return $S$;
```

Algorithm 1: Optimal strategy.

Table 2: Explanation for Algorithm 2's variables and functions.

Variable/function	Explanation
V	Vehicle set
(xcn, ycn)	Center coordinates of middle cell
$t1$	Last moment
$t2$	Current moment
d	Distance between service requester and service provider
Cset	A set of $\{\lambda_m, \theta_m, \lambda_f, r_{t,rsu}, v_{\text{vehicle}}, f\}$
pif	As explained in Table 1
Vtarget	The vehicle member currently traversed
get_position (V, t)	Get the position of vehicle V at time t
get_cellnum ((xt, yt), (xcn, ycn))	Obtain the cell number of coordinate (xt,yt) by taking coordinate (xcn, ycn) as a reference point
get_MSCN (cellnum)	Obtain the MSCN id corresponding to cell cellnum
get_rsu (MSCNid)	Obtain the RSU id to which MSCNid belongs
mscnMigration	A map < vehicle_id,flag > which represents the vehicle who's id is vehicle_id will perform or not service migration at the MSCN level while the flag is true or false
rsuMigration	A map < vehicle_id,flag > which represents the vehicle who's id is vehicle_id will perform or not service migration at the RSU level while the flag is true or false
setMigration	The service migration set {rsuMigration, mscnMigration}
get_strategy (pif, cset)	Obtain the optimal strategy according to Algorithm 1 with variable pif and variable Cse*t*

```
Input:
    $V$, $(x_{cn}, y_{cn})$, $t1$, $t2$, $d$, $Cset$, $p_{if}$
Output:
    setMigration {rsuMigration, mscnMigration}
for $V_{target}$ each in $V$
    $(x_{t1},y_{t1})$ ← get_position ($V_{target}$,t1);
    $(x_{t2},y_{t2})$ ← get_position ($V_{target}$,t2);
    cellnum1 ← getcell_num (($x_{t1}$, $y_{t1}$),($x_{cn}$, $y_{cn}$));
    cellnum2 ← getcell_num (($x_{t2}$, $y_{t2}$),($x_{cn}$, $y_{cn}$));
    MSCNid1 ← get_MSCN (cellnum1);
    MSCNid2 ← get_MSCN (cellnum2);
    rsu1 ← get_rsu (MSCNid1);
    rsu2 ← get_rsu (MSCNid2);
    if (MSCNid1 = = MSCNid2) then
        rsuMigration.insert ($V_{target}$, false);
    else
        mscnMigration.insert ($V_{target}$, true);
end
if (rsu1 = = rsu2) then
    rsuMigration.insert ($V_{target}$, false);
else
    S ← get_strategy ($p_{if}$,Cset);
if ($d > S$) then
    rsuMigration.insert ($V_{target}$, true);
    update $p_{if}$;
else
    rsuMigration.insert ($V_{target}$, false);
end
end
end for
Return setMigration;
```

Algorithm 2: Service migration under MSCN framework.

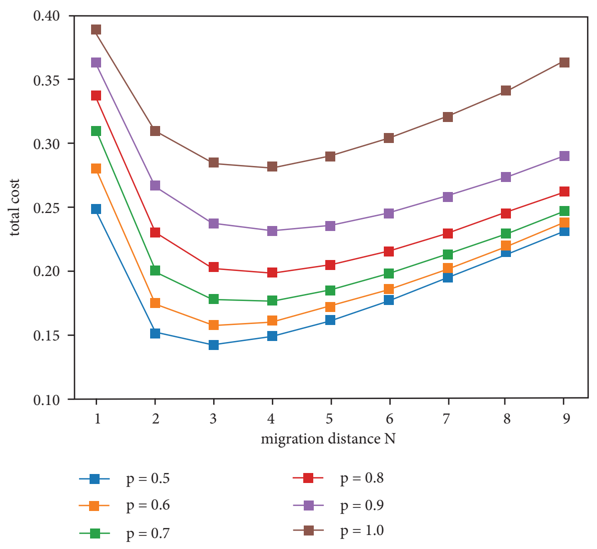


Figure 4: Total cost vs. migration distance N at different p.

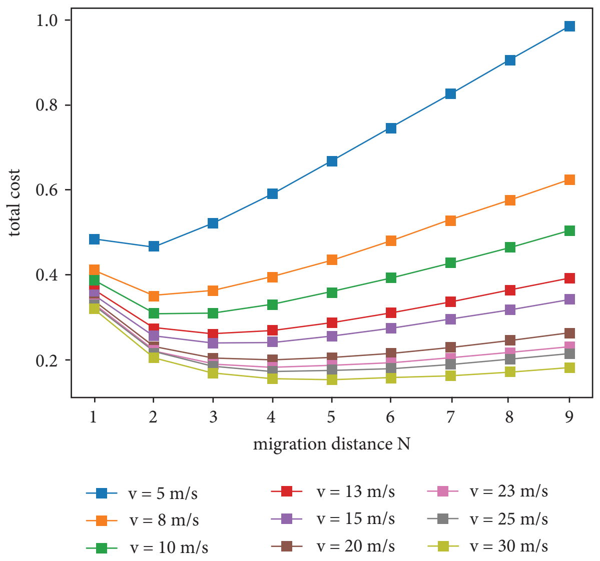


Figure 5: Total cost vs. migration distance (N) at different regional average speed (v).

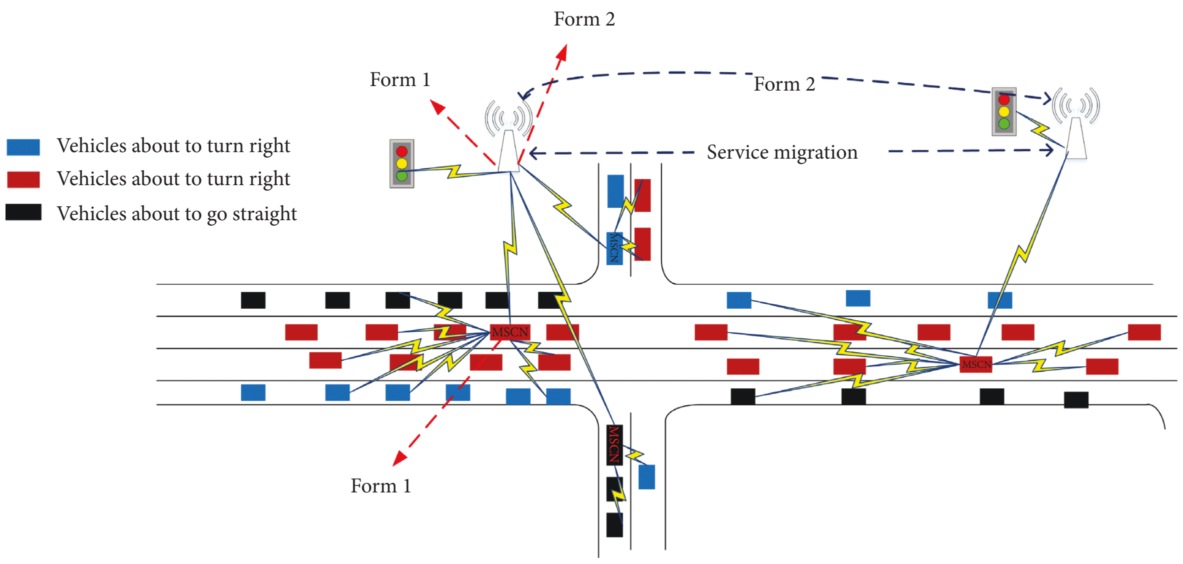

Form 2
Form 1
Form 2
Service migration
Vehicles about to turn right
Vehicles about to turn right
Vehicles about to go straight
MSCN
MSCN
MSCN
MSCN
Form 1

Figure 6: Intersection control under MSCN framework.

whether to perform service migration of applications based on DIMDP.

Intersection traffic flow detection in the MSCN framework requires to collect three sets of data, including

(1) The number of queued vehicles in each lane group (left-turn, straight turn, and right-turn lane group) of the main road at intersections.

(2) The number of vehicles queued on the branch entrance road.

(3) The speed of straight-line vehicles exiting the intersection, which can be obtained by the vehicle's own speed sensor.

The intersection control application in the MSCN framework is shown in Figure 6, in which blue vehicles represent upcoming right-turn vehicles after passing through the intersection, red represents upcoming straight-traffic vehicles, and black represents upcoming left-turn vehicles. RSUs deploying MEC services are set up between upstream and downstream intersections, and the distance between RSUs is determined according to their transmission range. The specific data flow is as follows:

(1) The road vehicles decide whether to perform service migration according to the DIMDP in the MSCN framework.

(2) The road vehicles send application messages (including node id, MSCN id, direction information, vehicle current speed, vehicle current acceleration, and time stamp information) to the MSCN in the region through the information distribution mechanism in the MSCN framework.

(3) The MSCN receives road vehicle information and performs preliminary calculations and statistics to obtain Table 3.

(4) The MSCN decides whether to perform application service migration based on the DIMDP optimization policy.

(5) The MSCN sends Table 3 to the optimal RSU.

(6) Optimal RSU assembles and organizes Table 3 from each cluster and MSCN to obtain Table 4.

(7) The adjacent RSUs exchange Table 4 to generate updating Table 4.

(8) The optimal RSU sends updating Table 4 to the traffic light controller and executes intersection control.

Table 3 shows the preliminary results after MSCN collects all the normal vehicle nodes and its own application information in its region and conducts statistical sorting. Then, the MSCN sends Table 3 to the optimal RSU, which receives Table 3 from each cluster and each MSCN within the cluster, processes, and counts it to form Table 4.

5. Simulation and Result Analysis

5.1. Simulation Setup. Simulation is done based on the NS-3.28 platform. Simulation parameters are listed in Table 5. We excerpted a 600-meter two-way road used in the simulation as a diagram as shown in Figure 7.

Corresponding communication parameters are listed in Table 6.

Note here that V2X information is one kind of timeliness information. Hence, re-transmission procedure is disabled. Moreover, until today, most C-V2X-based modules do not support power adjusting function and use fixed transmit power, valued as 23 dBm.

Three parameters, average backhaul delay (ABL), packet delivery rate (PDR), and effective feedback ratio (EFR), are used to evaluate the performance of the proposed mechanism.

Here effective feedback ratio is defined as the ratio of received effective service message number to uploaded BSM number, while the average backhaul delay is defined as the service response delay, which is the interval between the source node's BSM sending time t_{bsm} and destination node's service response time t_{service}. Considering the influence of BSM dissemination delay, here we employ the

Table 3: MSCN.

Node id	MSCN id	Sending time	Direction	Speed (m/s)	Acce (m/s2)
1	1	10.23.33	Right	20.5	1.01
8	1	10.23.45	Left	21.2	0.32
3	1	10.23.55	Right	24.3	1.51
2	1	10.23.58	Straight	23.5	0.58
. . .	. . .	. . .	. . .	. . .	. . .

Table 4: RSU.

Cluster id	MSCN id	Receiving time	Total right	Total left	Total straight
1	3	10.32.08	8	3	12
1	1	10.32.13	4	6	8
1	2	10.32.45	5	1	5
3	3	10.32.58	4	6	16
. . .	. . .	. . .	. . .	. . .	. . .

Table 5: Simulation parameters.

Parameter	Value
Lane number	2
RSU number	5
MSCN number	15
Distance between RSUs	300m
Lane width	3m
Lane length	1500m
Vehicles' velocity	± 20 m/s– ± 25 m/s

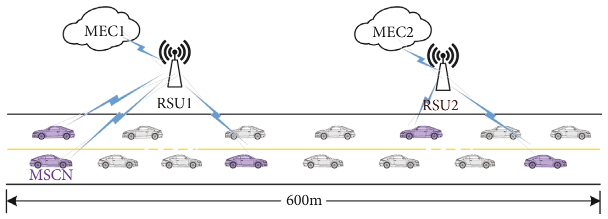


Figure 7: The diagram of the two-way road used in the simulation.

Table 6: Communication simulation parameters.

Parameter	Value
Frequency band	5.9 GHz
Channel bandwidth	10 MHz
Data rate	5 Mbps
Transmit power	23 dBm
Packet size	1200 bit
Packet delivery frequency	10 Hz

average BSM dissemination delay to calculate ABL t_{ABL} as follows:

$$t_{\mathrm{ABL}} = \frac{1}{N}\sum_{i=0}^{N} t_{\mathrm{service}} - t_{bsm}, \tag{21}$$

where N is the BSM packet number.

To verify the performance of DIMDP under MSCN architecture, three others are selected as a contrast.

(i) MDP service migration strategy without driving intention under MSCN architecture.

(ii) Always migration strategy under MSCN architecture, whose migration condition is set as $d(t) \geq 0$.

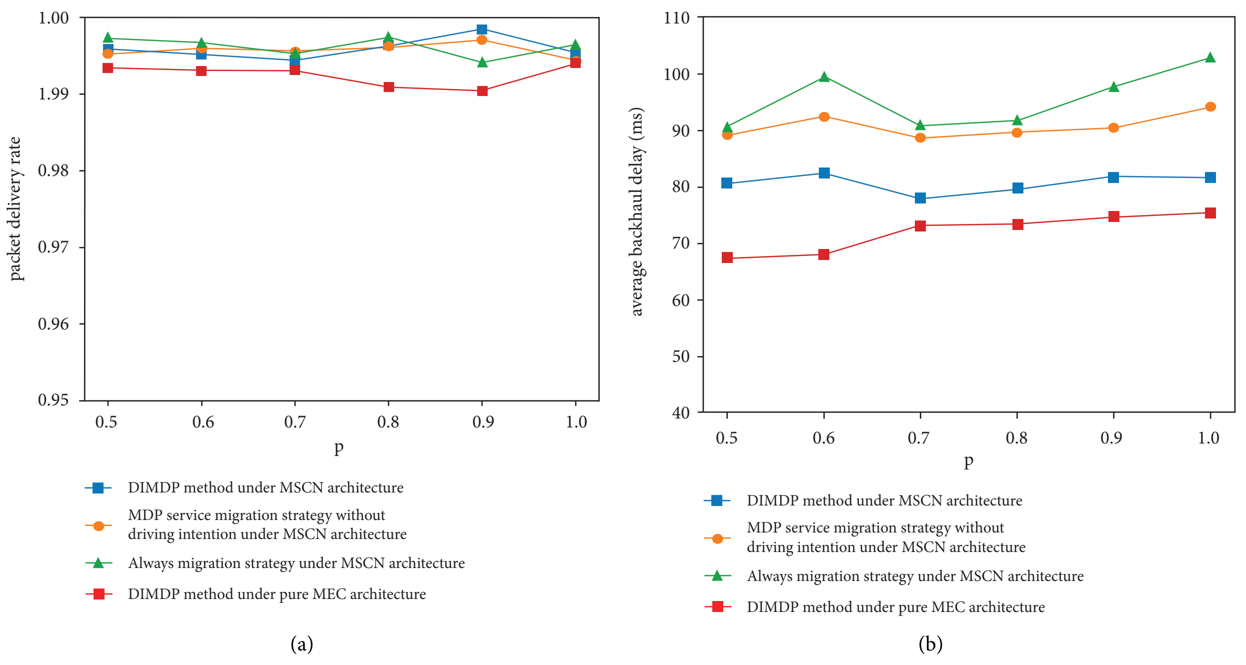


Figure 8: The impact of p value. (a) Packet delivery rate. (b) Average backhaul delay.

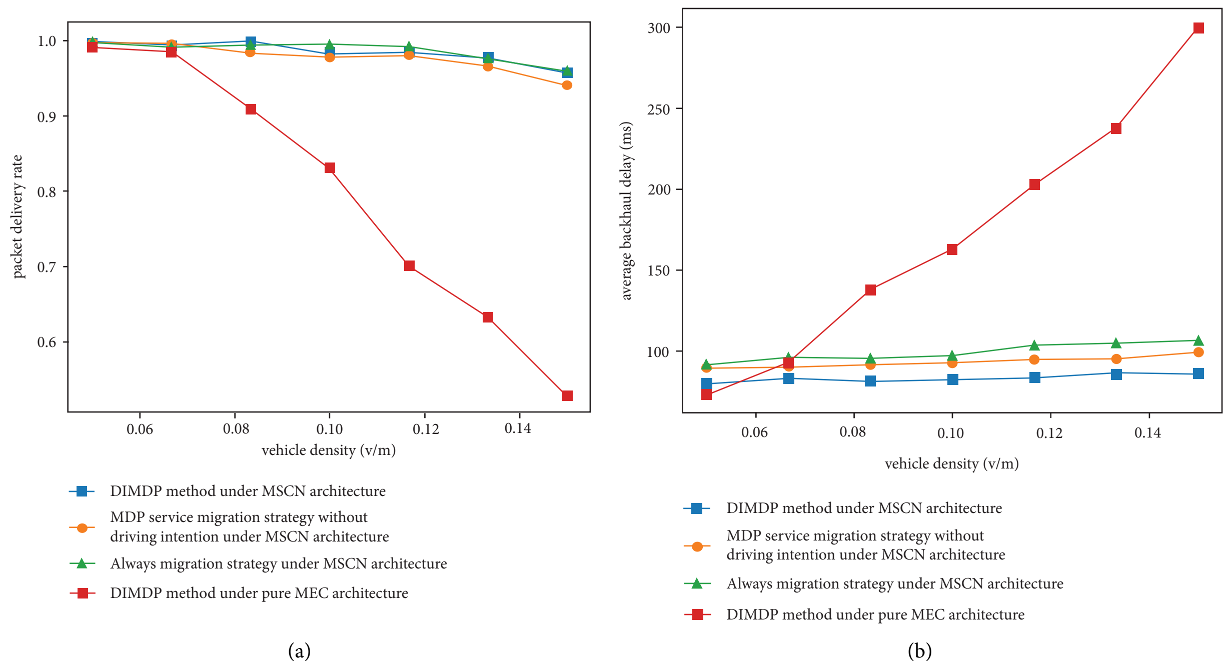


Figure 9: The impact of vehicle density. (a) Packet delivery rate. (b) Average backhaul delay.

(iii) DIMDP under pure MEC architecture.

As shown in Figure 8, the p value has no significant effect on packet delivery rate and backhaul delay. The proposed DIMDP method performs better backhaul delay, which means the V2X security applications can respond faster.

As shown in Figure 9, with the increase in road vehicle density, both the ABL and PDR of the proposed DIMDP method under MSCN architecture are nearly equal to a constant and perform better, which means it can still maintain excellent and stable performance in scenes with high vehicle density.

As shown in Figure 10, the higher the vehicle velocity is, the lower the effective feedback ratio should be. Moreover, the effective feedback ratio of the proposed DIMDP method under MSCN architecture is higher than that of other methods.

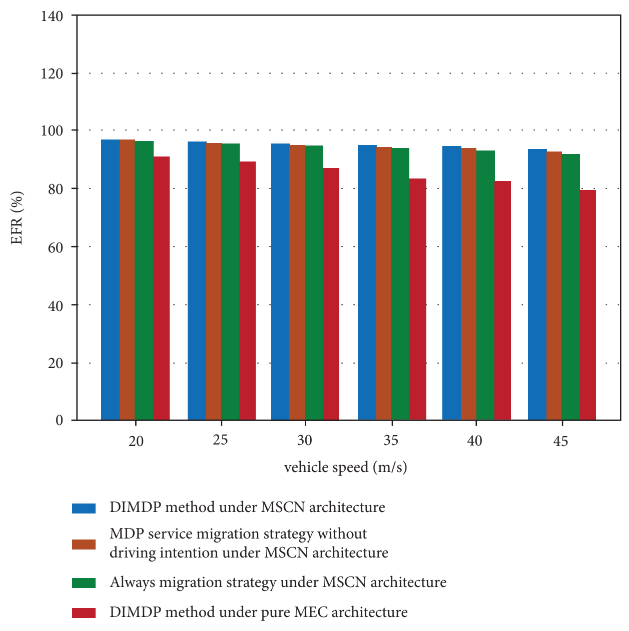


Figure 10: EFR vs. vehicle speed of four methods.

6. Conclusion

V2X safety applications have time-sensitive characteristics, and their application performance largely depends on the effectiveness of the information sharing issues. Due to the high mobility of vehicle nodes, service providers, such as MEC servers and cloud servers, could not disseminate the useful messages to vehicle nodes on time. Fortunately, a reasonable calculation framework and effective transmission guarantee measure are useful to solve the above problems.

MSCN framework, which uses public vehicles as mobile service providers, could decrease the service migration rate and maintain service continuity. On the other hand, although the MDP model could be used to express service migration progress, the vehicle motion features should be considered in the construction of the MDP model.

In this paper, we use vehicle driving intention and regional traffic conditions to express the vehicle's mobile tendency. A complex cost function, which considers both transmission factor and migration cost, is defined. Simulation results show that the proposed DIMDP model performs well.

In the future, we shall work on the vehicle motion model, thus further refining our model and improving its performance.

Data Availability

The simulation data supporting the system performance analysis can be obtained from https://github.com/lkw-sssy/DIMDP-A-Driving-intention-based-MDP-service-migration-model-under-MEC-MSCN-architecture.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the Special Key Project of Chongqing Technology Innovation and Application Development under grant no. csct2021jscx-gksbX0057.

References

[1] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: challenges and research directions," in *Proceedings of the Workshop on Mobile Edge Communications*, pp. 7–12, Los Angeles, CA, USA, August 2017.

[2] Q. Han, X. Zhanga, J. Zhang et al., "Research on resource scheduling and allocation mechanism of computation and transmission under MEC framework," in *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 437–442, IEEE, Auckland, New Zealand, October 2019.

[3] Q. Han, L. Lin, L. Zeng, J. Zhang, L. Ye, and K. Ling, "Research on service migration and typical applications under MSCN framework," in *Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference*, pp. 1023–1028, IEEE, Indianapolis, IN, USA, October 2021.

[4] X. Yu, M. Guan, M. Liao, and X. Fan, "Pre-migration of vehicle to network services based on priority in mobile edge computing," *IEEE Access*, vol. 7, pp. 3722–3730, 2019.

[5] S. Ge, M. Cheng, and X. Zhou, "Interference aware service migration in vehicular Fog computing," *IEEE Access*, vol. 8, pp. 84272–84281, 2020.

[6] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE," in *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Kuala Lumpur, Malaysia, May 2016.

[7] J. Xu, X. Ma, A. Zhou, Q. Duan, and S. Wang, "Path selection for seamless service migration in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9040–9049, 2020.

[8] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 1291–1296, Georgia, GA, USA, December 2013.

[9] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proceedings of the 2014 IEEE International Conference on Communications (ICC)*, pp. 1350–1354, IEEE, Sydney, Australia, June 2014.

[10] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: when cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 2019.

[11] M.-A. Lèbre, F. Le Mouël, and E. Ménard, "On the importance of real data for microscopic urban vehicular mobility trace," in *Proceedings of the 2015 14th International Conference on ITS Telecommunications*, pp. 22–26, ITST, Copenhagen, Denmark, December 2015.

[12] F. A. Silva, A. Boukerche, T. R. Silva, L. B. Ruiz, and A. A. Loureiro, "A novel macroscopic mobility model for vehicular networks," *Computer Networks*, vol. 79, pp. 188–202, 2015.

[13] O. Biham, A. A. Middleton, and D. Levine, "Self-organization and a dynamical transition in traffic-flow models," *Physical Review A*, vol. 46, no. 10, pp. R6124–R6127, 1992.

[14] J. Eggert and F. Mueller, "A foresighted driver model derived from integral expected risk," in *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1223–1230, Auckland, New Zealand, October 2019.

Hindawi
Mobile Information Systems
Volume 2022, Article ID 9230521, 11 pages
https://doi.org/10.1155/2022/9230521

Research Article

Joint Optimization for MEC Computation Offloading and Resource Allocation in IoV Based on Deep Reinforcement Learning

Jian Wang,[1] **Yancong Wang**,[1] **and Hongchang Ke**[2]

[1]*College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2]*School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun 130012, China*

Correspondence should be addressed to Hongchang Ke; kehongchang1981@163.com

Received 18 June 2022; Accepted 13 July 2022; Published 13 August 2022

Academic Editor: Konglin Zhu

In the Internet of Vehicle (IoV), the limited computing capacity of vehicles hardly processes the intensive computation tasks locally. The computation tasks can be offloaded to multiaccess edge computing (MEC) servers for processing, where MEC provides the required computing capacity to the nearby vehicles. In this paper, we consider a scenario where there are cooperation and competition between vehicles, the offloading decision of any vehicle will affect the decisions of the others, and the computing resource allocation strategies by MEC will dynamically change. Therefore, we propose a joint optimization scheme for computation offloading decisions and computing resource allocation based on decentralized multiagent deep reinforcement learning. The proposed scheme learns the optimal actions to minimize the total weighted cost which is designed as the vehicles' satisfaction based on the type of stochastic arrival tasks and dynamic interaction between MEC server and vehicles within different RSUs coverages. The numerical results show that the proposed algorithms based on decentralized multiagent deep deterministic policy gradient (DDPG) which is named De-DDPG can autonomously learn the optimal computation offloading and resource allocation policy without a priori knowledge and outperform the other three baseline algorithms in terms of the rewards.

1. Introduction

With the development of wireless communication technology and the rapid growth of vehicles, Internet of Vehicle (IoV) has become one of the most important applications of the Internet of Things (IoT) [1, 2]. However, due to the limitation of the computing resource of the vehicles, several tasks cannot be executed locally within the required delay [3]. To solve this problem, offloading IoV tasks to mobile edge computing (MEC) server is proposed as a feasible solution [4]. MEC is close proximity to the mobile vehicles, supplying more sufficient computation resource to the offloaded tasks [5, 6].

In recent years, many researches regarding MEC computation offloading in IoV have been studied [7, 8]. Some researchers have conducted to develop the optimization scheme in computation offloading under certain constraints, such as reducing the delay, computation resource overhead, and energy consumption [9–11]. Moreover, computation congestion that affects the performance of MEC server and the load balance of computation resource among MEC server have been considered into the computation offloading problem [12]. Although MEC server provides vehicles with the resource far beyond theirs, the resource of the MEC server may also be insufficient when massive vehicles access to MEC server simultaneously. Therefore, the rational resource allocation to optimize the performance of various objectives is also a significant issue of edge computing offloading [13–15]. Because of the high-speed mobility of vehicles and the randomness of tasks, as well as the cooperation and competition between vehicles in IoV, the computation resource allocation policy of MEC server based on different offloading decisions of vehicles has been discussed by many researchers. By jointly optimizing resource allocation and offloading strategy in IoV, the overall cost of computation resource, energy, and the delay is minimized in [16–18]. However, these methods require a large number of iterations to obtain a satisfied local optimum, which is not suitable for application scenarios where the environment changes rapidly and decisions need to be made in real time. Meanwhile, solving this type of optimization problem is usually nonconvex and NP-hard.

Deep reinforcement learning (DRL) which is the combination of deep learning (DL) and reinforcement learning (RL) can tackle the nonconvex optimization problem and has been widely used as an effective approach to optimize different issues including offloading decision-making and resource allocation strategy [14, 19–23]. The previous works make many efforts to optimize task offloading in IoV. For example, deep Q-network (DQN) is adopted in multiple vehicles offloading system to obtain the optimized offloading decisions which maximize the QoS of digital twinning-empowered IoV system [23]. Similar work proposes multiagent DQN-based computation offloading scheme, in which the uncertainty environment is considered so that the vehicles can make offloading decisions to achieve an optimal long-term reward [24]. A dynamic task offloading scheme based on Q-learning is implemented to minimize the delay, energy consumption, and total overhead in IoV system [25]. URLLC-aware task offloading algorithm based on deep Q-learning is studied to maximize the throughput of vehicles with satisfied constraints in [26]. Jointly considering the task priority, vehicles' service availability, and computation resource sharing incentive, an optimal offloading policy based on soft actor-critic (SAC) maximizes both expected reward and the policy entropy of the offloading tasks in the dynamic vehicular environment [27]. Moreover, DQN-based joint computation offloading and task migration optimization are applied to minimizing the total system cost in a 5G vehicle-aware MEC network [28]. The two-stage scheme is designed to joint optimization, where DQN is used in the first step to obtain the offloading strategy and deep deterministic policy gradient (DDPG) is utilized to generate the transmit power determination strategy of the vehicles [29]. None of the above researches consider the joint optimization of offloading strategy and computation resource allocation when multiple agents interact in a dynamic IoV environment.

Different from the existing works, we propose a decentralized multiagent deep reinforcement learning-based method to solve the joint optimization of computation offloading decision and resource allocation for MEC server in IoV. The objective of our work is to minimize the weighted cost of multiagent. In summary, our main contributions are as follows:

(1) We propose a IoV scenario supported by MEC server for dynamic task offloading decision and computation resource allocation in the environment with multiple RSUs cover multiple vehicles. In this cooperative scenario, because of the mobility of multivehicle and the stochastic arrival tasks, the computation offloading decision and resource allocated to multiple RSUs and multiple vehicles change in different time slots.

(2) Based on the proposed model, we consider both offloading decision-making and computation resource allocation to gain the minimum weighted cost, which is related to the end-to-end delay and computation resource cost. Moreover, we formulate the problem as a Markov decision process (MDP) and design the state, action, and reward functions.

(3) In order to effectively solve the abovementioned problem with continuous variables and meet the requirement of convergence, a joint optimization scheme based on decentralized multiagent DDPG (De-DDPG) is proposed. The simulation results show that the convergence of our proposed algorithm is verified and our proposed algorithm has better performance than other three baseline algorithms.

The remaining of this paper is organized as follows: In Section 2, an MEC framework with multiple RSUs and vehicles is introduced, and we construct the network model, communication model, and computation model. Section 3 describes the problem statement of the joint optimization. The solution based on decentralized multiagent DDPG (De-DDPG) is proposed in Section 4. In Section 5, the simulation results and analysis are presented. Finally, we conclude this paper in Section 6.

2. System Model

2.1. Network Model. A three-layer Internet of Vehicle (IoV) is considered in this paper (see Figure 1), which consists of an MEC server, M roadside units (RSUs), and N vehicles on a multilane road of length L.

The MEC server is connected to M RSUs via the fiber-optic link for receiving and transmitting the computation tasks. We assume that the total computing resource of MEC server is denoted as F. The RSUs denoted by $\mathscr{M} = \{1, 2, \ldots, M\}$ locate along the road with the same coverage range l. Therefore, we divide the road into M segments, and all vehicles are randomly and independently distributed in the segments with arrival rate λ. The RSU is responsible for forwarding messages between the MEC server and the vehicles. A set of vehicles periodically send messages to RSU within its communication range, which is denoted as $= \{1, 2, \ldots, N\}$. Vehicles have the same local computing capacity which is determined by the onboard unit (OBU) [30]. For each vehicle-i, it sends not only task messages but also its driving characteristics $\{p_i, v_i\}$, where p_i and v_i represent its 1-D position and speed, respectively. Here, we assume that the distances between vehicles follow the exponential distribution and the speeds of the vehicles are truncated Gaussian distributed, which is more appropriate for the actual situation of the road [31, 32]. In addition, we assume that each vehicle only processes one computation task within the current time period. The computation task of each vehicle is denoted as $T_i = \{C_i, D_i^{\text{in}}, D_i^{\text{out}}, t_i^{\max}\}$, where C_i is the required computation capacity to complete the task, D_i^{in} and D_i^{out} are the data size of the input and output for computing, respectively, and $t_i^{\max}$ is the maximum tolerable delay for the task completion. Vehicle needs to execute a computation task within a tolerable time period, and the task can be either processed locally or offloaded to the MEC server. We define the binary offloading strategy of vehicles as $\mathscr{X} = \{x_i | x_i \in \{0, 1\}, i \in N\}$, where $x_i = 0$ and $x_i = 1$ means

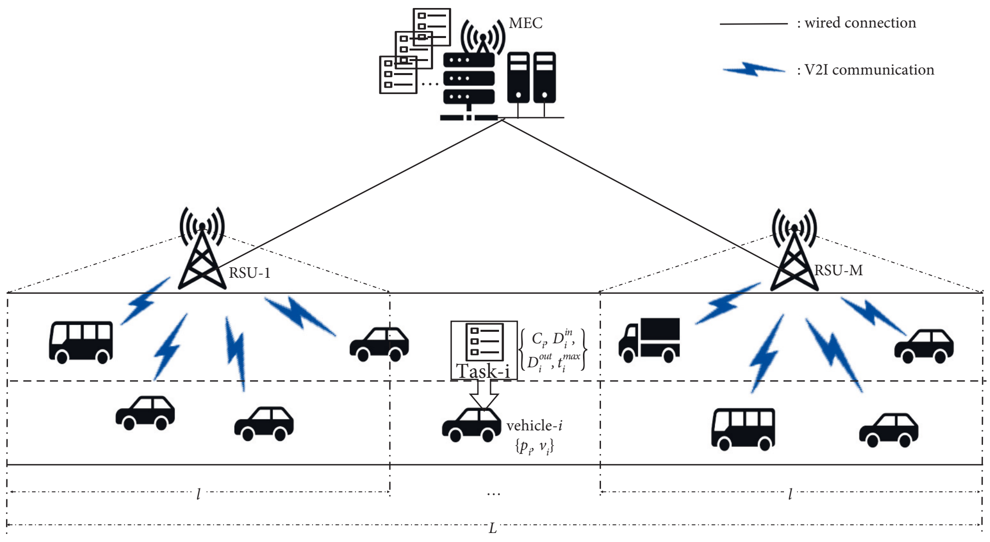


Figure 1: The system model of the MEC server and multivehicles.

that the vehicle-i decides to execute the computation tasks locally or offloads the task to the MEC server, respectively.

Moreover, when the vehicle leaves the coverage of the RSU, the vehicle will be disconnected from the RSU and can no longer transmit data to the MEC server through the RSU. The time available by the vehicle before leaving the communication range of RSU-j, $j = \lceil p_i/l \rceil, j \in M$, i.e., sojourn time, can be given as

$$t_{ij}^{soj} = \frac{l\lceil p_i/l \rceil - p_i}{v}, \qquad v = \frac{L\lambda}{N}, \tag{1}$$

where v represents the vehicle's equivalent speed and $\min\{v_i\} \le v \le \max\{v_i\}, i \in N$.

2.2. Communication Model. When the vehicles decide to offload the task to MEC server, the vehicles will transmit data to MEC server through the RSUs. Generally, the propagation time of the fiber-optic transmission between RSUs and the MEC server can be ignored [12]. We consider the V2I communication between the vehicle and the RSU is based on IEEE 802.11p in this work [33]. According to [34], the uplink and downlink transmitting rate $(r_{ij}^{UL}, r_{ij}^{DL})$ of the wireless communication between vehicle-i and its belonged RSU-j is expressed as

$$r_{ij}^{UL/DL} = \frac{D_i^{\text{in}}/D_i^{\text{out}} N_j \tau_{ij}\left(1-\tau_{ij}\right)^{N_j-1}}{\left(1-\tau_{ij}\right)^{N_j}\sigma + T_{ij}^{\text{success}} N_j \tau_{ij}\left(1-\tau_{ij}\right)^{N_j-1} + 1 - \left(1-\tau_{ij}\right)^{N_j} - N_j\tau_{ij}\left(1-\tau_{ij}\right)^{N_j-1}(\text{RTS}+\text{AIFS}+\delta)}. \tag{2}$$

where N_j is the number of vehicles which decide to offload task to MEC server via RSU-j. τ_{ij} represents the probability that vehicle-i connects to the RSU-j in a random time slot. σ is the duration of a time slot. RTS stands for request to send interval, AIFS denotes the arbitration inter-frame spacing interval, and δ expresses the propagation delay. T_{ij}^{success} is defined as the success transmission period between vehicle-i and RSU-j, which is written as

$$T_{ij}^{\text{success}} = \Phi + \frac{D_i^{\text{in}}/D_i^{\text{out}}}{\omega_j \log\left(1 + P_i h_{ij}\right)}, \tag{3}$$

where Φ is specific to the MAC protocol, and it equals $H + \text{SIFS} + \delta + \text{ACK} + \text{AIFS} + \delta + \text{RTS} + \text{SIFS} + \delta + \text{CTS} + \text{SIFS} + \delta$. $H = \text{PHY}_{\text{head}} + \text{MAC}_{\text{head}}$ represents the packet header's overhead. SIFS, ACK, and CTS stand for short interface space interval, acknowledgment interval, and CTS interval, respectively. ω_j denotes the bandwidth of RSU-j, P_i is the transmission power of vehicle-i, and h_{ij} stands for the channel gain between vehicle-i and RSU-j.

The uplink/downlink transmitting time under this situation is calculated as

$$t_{\text{UL/DL}}^{\text{mec}} = \frac{D_i^{in}/D_i^{\text{out}}}{r_{ij}^{\text{UL/DL}}}. \tag{4}$$

And the two-way transmission time between vehicle and RSU is given by

$$t_{\text{trans}}^{\text{mec}} = t_{\text{UL}}^{\text{mec}} + t_{\text{DL}}^{\text{mec}}. \tag{5}$$

2.3. Computation Model. The processing time is considered under two situations: the task is processed locally, and the task is offloaded to the MEC server for computing.

2.3.1. Local Processing Model. When vehicle-i processes its computation task locally ($x_i = 0$), the processing time of vehicle-i t_i^{loc} is only dependent on its own computing capacity. The local execution time $t_{\mathrm{exe}}^{\mathrm{loc}}$ is formulated as

$$t_i^{\mathrm{loc}} = t_{\mathrm{exe}}^{\mathrm{loc}} = \frac{C_i}{f_{\mathrm{loc}}}. \tag{6}$$

Here, f_{loc} is denoted as the vehicle's computation capacity, which is related to the vehicle's CPU cycle frequency.

2.3.2. MEC Processing Model. When the task is offloaded to the MEC server ($x_i = 1$), the end-to-end delay of vehicle-i includes the task execution time and the transmitting time. The execution time of vehicle-i offloading the task to MEC server is given as

$$t_{\mathrm{exe}}^{\mathrm{mec}} = \frac{C_i}{f_j^{\mathrm{mec}}}, \tag{7}$$

where f_j^{mec} denotes the computation capacity assigned to RSU-j which connects to vehicle-i by the MEC server, and f_j^{mec} denotes the allocated CPU cycle frequency of RSU-j by the MEC server. The end-to-end delay between vehicle-i and the MEC server is obtained by

$$t_i^{\mathrm{mec}} = t_{\mathrm{exe}}^{\mathrm{mec}} + t_{\mathrm{trans}}^{\mathrm{mec}}. \tag{8}$$

The main notations and descriptions are described in Table 1.

3. Problem Statement

In this section, the optimization problem is formulated by jointly considering the offloading decision and resource allocation with the aim of load balance and system cost minimization. First of all, we define the cost function as follows.

Cost function is considered to quantify the satisfaction level of the vehicle's offloading decision, which is inversely related to the satisfaction and identified by the delay sensitivity and the cost of computation resource. The logarithmic function is known as proportional fairness in many researches [35], which can achieve load balance, and a logarithm function is used to represent the cost function in this paper. The processing delay of a task is generally considered to be inversely proportional to the satisfaction; that is, the shorter the task processing delay, the higher the satisfaction. In addition, if the task is completed within the maximum tolerable delay, the satisfaction of the vehicle should be non-negative. But once the completion processing time of the task exceeds its maximum tolerable delay, the processing result of the task will lose its value because the tasks in IoV are extremely tolerant of delays. Here, the penalty mechanism is brought into consideration. Another metric in the cost function is the computation resource cost. It is necessary to pay for the vehicle's computation resource when the vehicle processes the task locally. Furthermore, when the task is offloaded to the MEC server, it takes the vehicle's corresponding cost for computation resources allocated by the MEC server, which will also reduce the satisfaction of the vehicle. Therefore, the cost function for vehicle-i to process the task locally is given by

$$U_i^l = \begin{cases} \beta \log\left(1 + \left(t_i^{\mathrm{actual}} - t_i^{\mathrm{loc}}\right)^+\right) + (1-\beta)\rho f_{\mathrm{loc}}, & t_i^{\mathrm{loc}} \le t_i^{\max}, \\ P, & t_i^{loc} > t_i^{\max}, \end{cases} \tag{9}$$

where $\beta \in (0,1)$ and $1-\beta$ represent the weights of delay and computation resource cost, respectively. The weighted function provides a flexible scheme for different applications' specific requirements by adjusting the weight parameters. $(z)^+ = \max(z, 0)$ ensures that U_i^l is non-negative. ρ is the unit cost of the computing resource. And, $P > 0$ represents the penalty for the task that is not completed within its maximum tolerable delay.

Similarly, the cost function of vehicle-i offloaded the task to the MEC server for processing and can be expressed as

$$U_i^{\mathrm{mec}} = \begin{cases} \beta \log\left(1 + \left(t_i^{\mathrm{actual}} - t_i^{\mathrm{mec}}\right)^+\right) + (1-\beta)\rho f_i^{\mathrm{mec}}, & t_i^{\mathrm{mec}} \le t_i^{\mathrm{actual}}, \\ P, & t_i^{\mathrm{mec}} > t_i^{\mathrm{actual}}. \end{cases} \tag{10}$$

Because when the vehicle leaves the coverage of the RSU, the vehicle will disconnect to the MEC server through the RSU regardless of whether the task is processed or not. $t_i^{\mathrm{actual}} = \min\left\{t_{ij}^{soj}, t_i^{\max}\right\}$ is used to depict the task's actual tolerant delay.

Combining equations (9) and (10), the cost function of vehicle-i can be expressed as

$$U_i = \begin{cases} U_i^{\mathrm{loc}}, & \text{if } x_i = 0, \\ U_i^{\mathrm{mec}}, & \text{if } x_i = 1. \end{cases} \tag{11}$$

This work aims to minimize the system cost by jointly determining the offloading decisions of vehicles and the computation resource allocation of the MEC server. The optimization problem is formulated as

$$\begin{aligned} &\min_{\mathcal{X},\mathcal{F}} \sum_{i=1}^{N} U_i, \\ &s.t. C1\colon 0 \le f_{\mathrm{loc}} < F, \\ &C2\colon 0 \le f_i^{\mathrm{mec}} \le x_i F, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}, \\ &C3\colon \sum_{j=1}^{M} f_i^{\mathrm{mec}} \le F, j \in \mathcal{M}, \\ &C4\colon x_i = \{0,1\}, i \in N. \end{aligned} \tag{12}$$

Constraint C1 ensures that the available local computation resource is non-negative and less than the MEC server. C2 is the constraint of the available computation resource assigned for each vehicle-i within the coverage of RSU-j by the MEC server. The sum of the computation resource allocated to all the offloading tasks through RSU-j does not exceed the total computation resource of the MEC

Table 1: Notation description.

Notation	Description
L	The length of the selected road
l	The coverage range of the RSU
$\mathscr{M}, M$	Set/number of RSUs
$\mathscr{N}, N$	Set/number of vehicles
i,j	The vehicle index $i \in \mathscr{N}$/the RSU index $j \in \mathscr{M}$
λ	The arrival rate of vehicles
p_i, v_i	Vehicle's position/speed
$C_i, D_i^{\text{in}}, D_i^{\text{out}}, t_i^{\max}$	Required computation resource/input data size/output data size/maximum tolerable delay of the computation task T_i
t_{ij}^{soj}	The time available by the vehicle before leaving the communication range of RSU--j
v	The equivalent speed of vehicles
r_{ij}^{UL}, r_{ij}^{DL}	The available uplink/downlink transmission rate of vehicle-i
N_j	The number of vehicles offloads task to MEC server via RSU-j
τ_{ij}	The probability of vehicle-i connects to the RSU-j in a random time slot
σ	The duration of a time slot
δ	The propagation delay
T_{ij}^{success}	The success transmission period between vehicle-i and RSU-j
ω_j	The bandwidth of RSU-j
P_i	The transmission power of vehicle-i
h_{ij}	The channel gain between vehicle-i and RSU-j
$t_{\text{UL}}^{\text{mec}}, t_{\text{DL}}^{\text{mec}}$	The uplink/downlink transmitting time
$t_{\text{trans}}^{\text{mec}}$	The two-way transmission time between vehicle and RSU
x_i, $\mathscr{X}$	The binary offloading strategy of vehicle-i/vehicles
$t_{\text{exe}}^{\text{loc}}$, $t_{\text{exe}}^{\text{mec}}$	The task execution time locally/in the MEC server
t_i^{loc}, t_i^{mec}	Total time for processing task T_i locally/in the MEC server
f_{loc}, f_j^{mec}, F	Computing resource of the vehicle/allocated to MEC-j/the MEC server
U_i^l, U_i^{mec}, U_i	The cost of vehicle-i locally/in MEC processing/vehicle-i under different task offloading decisions
P	The penalty for offloading failure
β	The weighted parameters of delay and computation resource cost
t_i^{actual}	The task's actual tolerant delay
ρ	The unit cost of the computing resource of the MEC server

server in the constraint C3. C4 shows the binary offloading decision constraint for vehicle's task.

Since the cost function in the above problem involves the end-to-end delay, which is related to the indicators of the stochastic arrival tasks $C_i, D_i^{\text{in}}, D_i^{\text{out}}$, the computing resource is allocated to RSU-j f_j^{mec} and the relative position of vehicle to RSU based on vehicle's driving characteristics p_i, v_i. Therefore, the computation complexity is an additive change on all of tasks and the vehicle characteristics. In addition, the computation complexity also depends on the number of the generated tasks. In this optimal problem, the offloading decisions $\mathscr{X}$ and the allocated computation resource $\mathscr{F}$ are two main challenges which make the problem into a mixed-integer nonlinear programming problem that is generally nonconvex and NP-hard [36]. We adopt a multiagent deep reinforcement learning approach to feasibly solve the problem of jointly optimizing the computation offloading decision and computation resource allocation.

4. DRL for Computation Offloading and Resource Allocation

4.1. Scheme Design. We assume that the state is determined by the arrival tasks and the vehicle's characteristics which are updated in each step. The state of the next time slot is related to the state of the current time slot. Therefore, the formulated problem can be modeled as a Markov decision process (MDP). MDP is the iterative process in which agents observe the states in state space from the environment, select an action from action space, obtain an immediate reward sequentially, and then transit to another state, which can be represented as a tuple $<S, A, P_{s,a}, R, \gamma>$, where S is state space, A is action space, $P_{s,a}$ is transition probability space, R is reward space, and γ is discount factor. MDP policy is completely dependent on the current state. The state space is designed to accommodate the proposed IoV environment. Each vehicle acts as the agent. At first, we define the state space, action space, and reward space as follows.

4.1.1. State Space. The state at time slot t is corresponding to the required computation capacity to complete the task C_i, the input data size of the task D_i^{in}, the output data size of the task D_i^{out}, the position of vehicle p_i, the speed of vehicle v_i, and the computing resource allocated to RSU-j. Thus, the state $s_i(t)\epsilon S$ can be described as

$$s_i(t) = \left\{C_i(t), D_i^{\text{in}}(t), D_i^{\text{out}}(t), p_i(t), v_i(t), f_j^{\text{mec}}(t)\right\}_{\forall i\in\mathscr{N},\forall j\in\mathscr{M}}. \tag{13}$$

4.1.2. Action Space. The action is the joint decision-making for the computation offloading and the resource allocation. The vehicle needs to decide to process the task locally or offload to the MEC server. If the task is offloaded to the MEC server, the computation resource is allocated to the vehicle by the MEC server via the linked RSU. Therefore, the action $a_i(t)\epsilon A$ is composed of the binary offloading decision and the computation resource allocated to vehicle-i, depicted as

$$a_i(t) = \left\{ \left\{ x_i(t), f_i^{\mathrm{mec}}(t) \right\}_{\forall i \in \mathcal{N}} \right\}. \tag{14}$$

4.1.3. Reward Space. We assume that all vehicles with the same functionality can share the same reward function. Each agent selects its action based on the reward to obtain the maximum global reward. The long-term weighted sum of the cost function of all the tasks is considered as the objective, and we define the below function to maximize the reward function (minimize the cost function) during the whole time period T.

$$R_i(t) = -\lim_{T \longrightarrow \infty} \frac{1}{T} \sum_{t=1}^{T} U_i(t) | s_i(t). \tag{15}$$

The average rewards of all agents in time slot t can be calculated as

$$R(t) = \frac{1}{N} \sum_{i=1}^{N} R_i(t), \forall i \in \mathcal{N}. \tag{16}$$

Minimizing the weighted cost function of the proposed model amounts to maximizing the average cumulative reward. The expectations of future rewards can be used to measure whether the selected action is appropriate or not. The reward is the return of the selected action based on the state in time slot t. Therefore, the cumulative reward which is generally indicated as the weighted expectation is maximized to select the optimal actions, formulated as

$$Q^{\pi}(s(t), \pi(t)) = \mathbb{E}\left[R(s(t), a(t)) + \gamma Q^{\pi}(s(t+1), \pi(t+1)) \right], \tag{17}$$

where $\gamma \in [0, 1]$ is the discount factor, π is the policy, and π^* is the optimal policy. $Q^*(s, a) = \max_{\pi^*} Q^{\pi}(s, \pi)$ corresponds to the agents' optimal policy $\pi^* = \{\pi_1^*, \pi_2^*, \ldots, \pi_N^*\}$.

4.2. Optimal Scheme Based on Decentralized DDPG. After formulating the MDP, we propose the optimization strategy based on the decentralized multiagent DDPG (De-DDPG) in this subsection, in which each agent is initialized with four deep neural networks (DNNs): the critic network, the actor network, and two copies of the actor and critic networks as target networks, respectively (see Figure 2) [37]. Each agent's state, action, and reward are obtained and used to train the DNNs during the training procedure. After training, each agent can select the next step strategy by its own actor network according to the local observation from the environment.

As shown in Algorithm 1, the process of De-DDPG algorithm can be divided into three parts: initialization, interaction, and update. At the beginning of the algorithm, four networks of each agent and the replay buffer B are initialed, where the critic network is $Q(s, a|\theta_i^Q)$, the actor network is $\mu(s|\theta_i^{\mu})$, the target critic network is $Q'(s, a|\theta_i^{Q'})$, and the target actor network is $\mu'(s|\theta_i^{\mu})$, respectively. In addition, the replay buffer can be large because the proposed De-DDPG is an off-policy algorithm, which allows the algorithm to benefit from learning across a set of uncorrelated transitions [38]. In the interaction procedure, for each episode, a sampled noise from the random noise process $\mathcal{N}_t$ is added to an exploration policy μ' into the actor policy. The reason for introducing random noise is to solve the problem of insufficient exploration of the environment by the output actions in deterministic policy algorithms. The Ornstein–Uhlenbeck process is used to generate temporally correlated exploration for exploration efficiency. Then, the actions interact with the environment and obtain the corresponding rewards and the next step states. According to the observation, the transitions $(s_i(t), a_i(t), R_i(t), s_i(t+1))$ store in the replay buffer B. When updating, a random mini-batch of Z transitions is sampled from the replay buffer. Then, update each agent's critic network, actor network, and two target networks in turn. Loop through each episode until the algorithm ends. In the update process, the critic network $Q(s, a|\theta_i^Q)$ is updated by minimizing the loss $L(\theta^Q)$ in Algorithm 1 which is the approximation function of other agent policy by each agent. Here, y_i is the predication of the next action in target actor network in formula (17). The actor network $\mu(s|\theta_i^{\mu})$ is updated by using the sampled policy gradient $\nabla_{\theta^{\mu}} J$ in Algorithm 1 which is the unbiased estimation of the policy gradient expectation calculated by the mini-batch transitions according to Monte Carlo method. After training the mini-batch transitions and updating the weights of critic network and actor network (θ_i^Q and θ_i^{μ}), the weights of two target networks for each agent ($\theta_i^{Q'}$ and θ_i^{μ}) can be soft updated as a running average algorithm, which is shown in Algorithm 1, respectively.

5. Numerical Results

This section describes the comprehensive numerical simulation analysis from simulation setup, simulation comparison, and simulation results.

5.1. Simulation Setup. Firstly, we evaluate and verify our proposed De-DDPG by TensorFlow 1.13.1. A personal computer with a RTX2070 GPU and 8GB video memory is used to train and test De-DDPG.

There are 20 vehicles driving on the road, 4 RSUs are located at the stationary region on the roadside, and an MEC server directly connects with RSUs. That is to say, 2 groups of RSUs are set to serve all vehicles. One RSU group includes the main RSU and a secondary RSU, and the purpose is to prevent the main RSU from being abnormal due to accidents (such as power failure and communication blockage). Therefore, $N = 20$, $M = 4$. The required computation resource is set as $C_i = 0.54$ (G cycles/Mbits). The size of arrival tasks follows uniform distribution $D_i^{\mathrm{in}} \sim U(1.6, 4.6)$ MetaBits. The arrival probability of tasks is 0.45. The maximum tolerable delay of the computation task is set as $t_i^{\max} = 4$ time

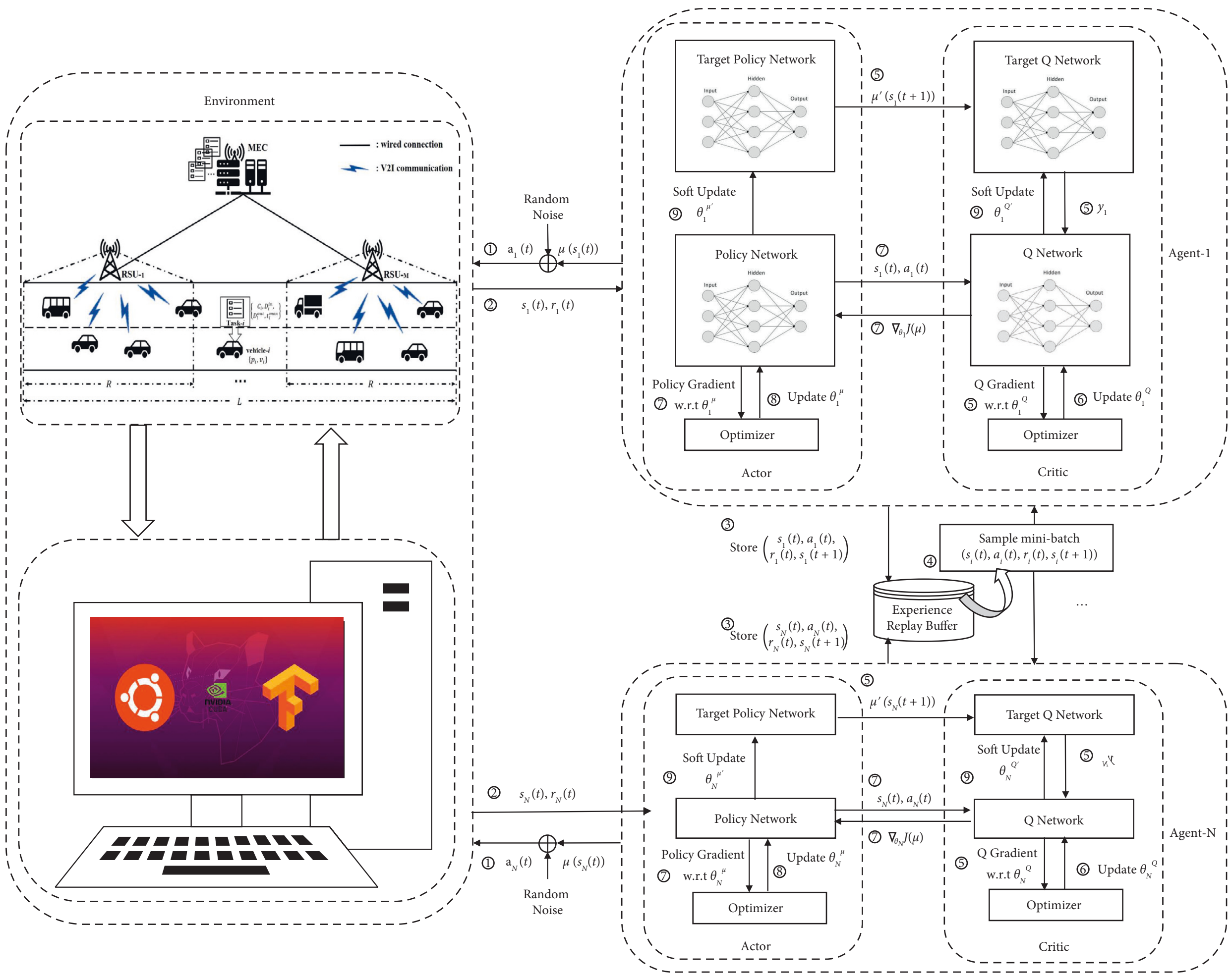


FIGURE 2: Structure of the proposed multiagent DDPG (De-DDPG) scheme.

slots. One time slot $t \in T$ is set to $t = 1$ ms. The uplink transmission rate of vehicle-i r_{ij}^{UL} is between 0.85 and 0.95 Mbits per time slot. The bandwidth of RSU $\omega_j = 10$ MHz. The transmission power for vehicle-i is $P_i = 1W$. The max computation resource allocated to RSU-j is 3.8 gigacycles. The computation resource of each vehicle is 0.5 gigacycles.

In terms of each vehicle-i ($i \in N$), the neural networks (two actor networks and two critic networks) for De-DDPG are composed of four layers, i.e., an input layer and two fully connected layers. Table 2 illustrates the parameters and values.

5.2. Simulation Comparison.

For verifying the performance of the proposed DE-DDPG, three benchmark algorithms are set: centralized deep deterministic policy gradient (Ce-DDPG), all tasks offloaded to RSU(A-RSU), and all tasks executed by local processor (A-LP), which are described as follows:

(1) Ce-DDPG: on the MEC side, a centralized controller captures global information such as tasks generated from vehicles, computation, and communication resources of RSUs. That is to say, there is only one agent which interacts with the MEC environment. In order to improve the convergence of Ce-DDPG, the allocated computation resources by all RSUs are the same. The structure of the neural network is the same as each sub-network of De-DDPG.

(2) A-RSU: all tasks from vehicles are offloaded to RSU in the corresponding coverage region. Furthermore, the computation resources allocated for each vehicle are the same.

(3) A-LP: all tasks are executed by the local processor of the vehicle.

5.3. Simulation Results.

In this section, we analyze the simulation results in detail from the two aspects: the convergence of proposed De-DDPG and the advantages of De-DDPG compared to three baseline algorithms.

In terms of De-DDPG's convergence, the average cumulative reward of De-DDPG in one period (the whole time slots) for each episode is formulated.

Figure 3 shows the convergence of proposed De-DDPG with different critics' learning rate α. The choice of learning rates α can obviously affect the convergence effect and speed of De-DDPG. From Figure 3, it can be observed that De-DDPG cannot be convergent when $\alpha = 0.01$. However, from

```
Randomly initialize critic network $Q(s,a|\theta_i^Q)$ and actor $\mu(s|\theta_i^Q)$ with weights $\theta_i^Q$ and $\theta_i^Q$
Initialize target network $Q'$ and $\mu'$ with weights $\theta_i^{Q'} \leftarrow \theta_i^Q$, $\theta_i^{\mu'} \leftarrow \theta_i^\mu$
Initialize replay buffer $B$
for for episode $k = 1, 2, \ldots, K$ do
  Initialize a random process $N_s$ foe action exploration
  Receive initial observation state $\mathbf{S} = \{s_i(1), s_i(2), \ldots s_i(N)\}$
  for $i = 1, 2, \ldots, N$do
    for $t = 1, 2, \ldots, T$do
      Select action $a_i(t)\mu(s_i^t|\theta_i^\mu) + N_s$ according to the current policy and exploration noise $N_s$
      Execute action $a_i(t)$ and observe reward $R_i$ and observe the next state $s_i(t+1)$
      Store all transitions $(s_i(t), \mathbf{a}_i(t)R_i(t), s_i(t+1))$ in $B$
      Sample a random mini-batch of $Z$ transitions $\{s_i(t), \mathbf{a}_i(t)R_i(t), s_i(t+1)\}$ from $B$
      Set
        $y_i = R_i(t) + \gamma Q_\pi(s_i(t), \mathbf{a}_i(t)|\theta^Q)|_{a_i^{k+1'} = \mu'(s_i^{k=t})}$
      Update critic network $Q(s, \mathbf{a}|\theta^Q)$ by minimizing the loss
        $L(\theta^Q) = 1/Z\sum_I (y_i - Q_\mu(s_i, \mathbf{a}_i(t)|\theta_i^Q))^2$
      Update the actor policy by using the sampled policy gradient
        $\nabla_{\theta_i^a} J \approx 1/Z\sum_i \nabla_a Q(s_i(t), \mathbf{a}_i(t)|\theta_i^Q)|_{a_i=\mu(s_i)} \nabla_{\theta_i^\mu} \mu(s_i(t)|\theta_i^\mu)$
      Update the target networks for each agent $i$:
        $\theta_i^{Q'} \leftarrow \eta\theta_i^Q + (1-\eta)\theta_i^{Q'}$,
        $\theta_i^{\mu'} \leftarrow \eta\theta_i^\mu + (1-\eta)\theta_i^{\mu'}$.
    end for
  end for
end for
```

Algorithm 1: Decentralized multiagent DDPG optimization method.

Table 2: Main hyperparameters of the De-DDPG.

Parameters	Value
Size of the first hidden layer for actor and critic	300
Size of the second hidden layer for actor and critic	300
Learning rate of actor and critic α'/α	0.0001/0.001
Size of experienced memory B	20000
Parameters for OU noise θ, μ, σ	0.15, 0.15, 0.10
Discount factor γ	0.95
Penalty for failed task execution P	8
Total number of all episodes K	1000
Total time periods of one episode T	110

the blue curve when $\alpha = 0.0001$, although De-DDPG can be eventually convergent, the convergence speed is too slow to influence the performance of De-DDPG. Therefore, we set $\alpha = 0.001$ because De-DDPG is more stable. In terms of actor's learning rate, we set $\alpha' = 0.0001$.

f_j^{mec} is the total computation resources preallocated to RSU-j from the MEC server. To ensure the fairness of computation resources allocation and the robustness of the proposed De-DDPG, we periodically update the allocated computation resources of each RSU by adding the fluctuation volatility rate. As shown in Figure 4, when the volatility rate is set to 1, 3, 5, the convergence and performance of De-DDPG are different. When $\lambda_j = 1$, the performance of De-DDPG is better than the other two curves. However, the stability of De-DDPG is slightly worse. As the volatility rate increases, the performance of De-DDPG decreases, but we can see that when the volatility rate is set to 3, the stability and convergence of the De-DDPG are optimal.

Figure 5 reveals the convergence of De-DDPG with different control parameters ρ. From formula (10), the greater the ρ is, the greater the cost is (the less the reward is). Although the average cumulative rewards of De-DDPG are worst when $\rho = 0.08$, the stability is much better than that of the curves when $\rho = 0.04$ and $\rho = 0.06$. From the curves shown in Figure 5, as the control parameter ρ increases, the performance of De-DDPG declines less and less. However, when $\rho = 0.08$, the convergence effect and training speed of De-DDPG are obviously improved. Therefore, we set the control parameter for the cost of computation resource $\rho = 0.08$ in this paper.

Furthermore, Figures 6–8 will verify the performance and advantages of the proposed De-DDPG compared to other baselines Ce-DDPG, A-RSU, and A-LP. The average cumulative reward of De-DDPG for all episodes (1000 episodes) is introduced to show the performance of four algorithms.

Figure 6 illustrates the comparison of four algorithms with the different numbers of arrival vehicles. When the number of vehicles N within the coverage region of each RSU is 6 or 8, because the computation resources of each RSU are sufficient to meet the computational demands of the vehicles for all generated tasks, the performance of De-DDPG, Ce-DDPG, and A-RSU is not significantly different. In terms of A-LP, regardless of the number of vehicles, its local processing ability remains unchanged, and the computation resources of RSUs have no effect on its performance. On the contrary, because the computation ability of the local processor for vehicles is insufficient for arrival tasks, a large number of penalties P in each episode will be incurred due to the incomplete tasks, thus affecting the average

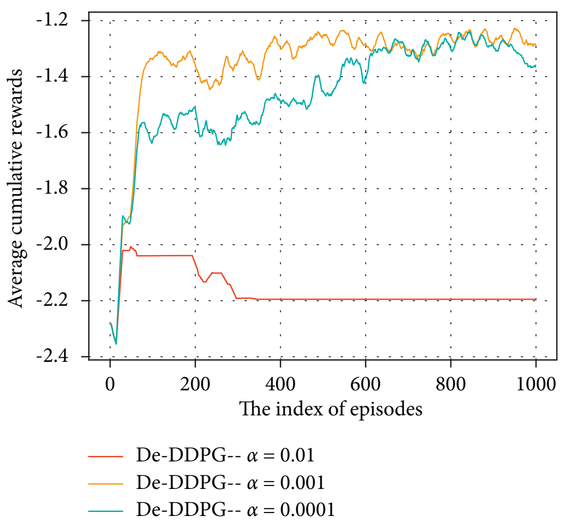


FIGURE 3: Convergence of proposed De-DDPG with different learning rates α.

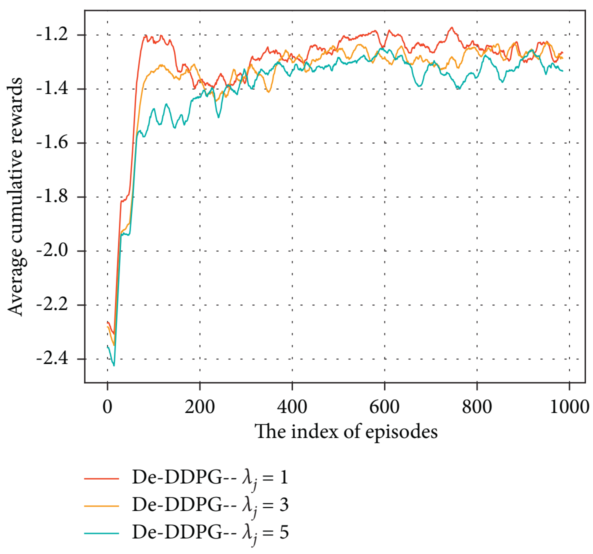


FIGURE 4: Convergence of proposed De-DDPG with different volatility rates λ_j.

cumulative reward of A-LP. However, when the number of vehicles N within the coverage region of each RSU is greater than 10, the computation resources of RSUs are insufficient for completing all tasks generated by all vehicles, and the performance of four algorithms degrades significantly. From the curves, the performance of the proposed De-DDPG is better than that of the other three algorithms.

The uplink transmission rate of the wireless communication between vehicle-i and its belonged RSU-j r_{ij}^{UL} can influence the performance of four algorithms. Figure 7 shows the performance of four algorithms with different uplink transmission rates r_{ij}^{UL}. We can see that A-LP is not influenced by the transmission rate r_{ij}^{UL} because A-LP executes all tasks by its own local processor. For De-DDPG, Ce-DDPG, and A-RSU, different uplink transmission rates r_{ij}^{UL} mean different transmission delays $t_{\mathrm{UL}}^{\mathrm{mec}}$ which affairs the

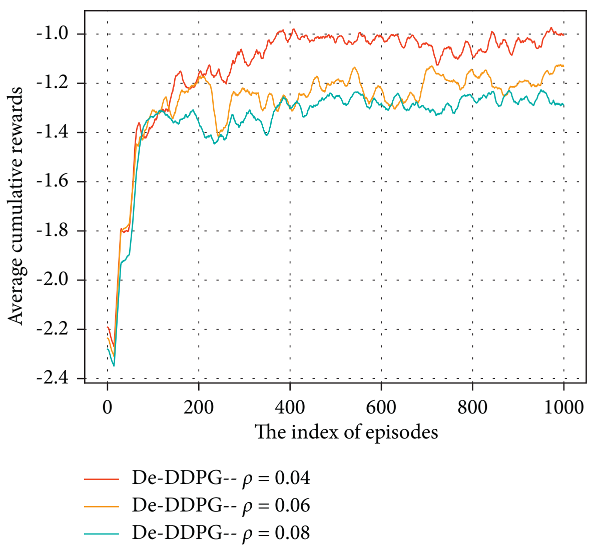


FIGURE 5: Convergence of proposed De-DDPG with different control parameters ρ.

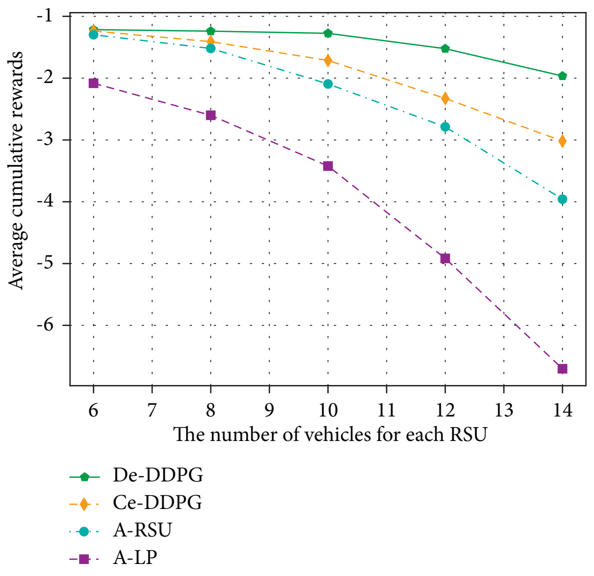


FIGURE 6: Comparison on all algorithms with the number of vehicles N.

average cumulative rewards of all three algorithms. As shown in Figure 7, the performance of De-DDPG is obviously better than that of Ce-DDPG and A-RSU due to the number of agents participating in training, offloading decisions, and the ratio of resource allocation.

In Figure 8, we describe the comparison of four algorithms with different trade-off coefficients β for latency cost. From formula (10), β is the weighted parameter of delay cost and $1-\beta$ computation resource cost. In terms of A-LP, since the computational resource cost is fixed, the performance of A-LP decreases as the trade-off coefficient β increases. However, the other three algorithms consider the trade-off between delay cost and computation resource cost, so the performance of De-DDPG, Ce-DDPG, and A-RSU varies

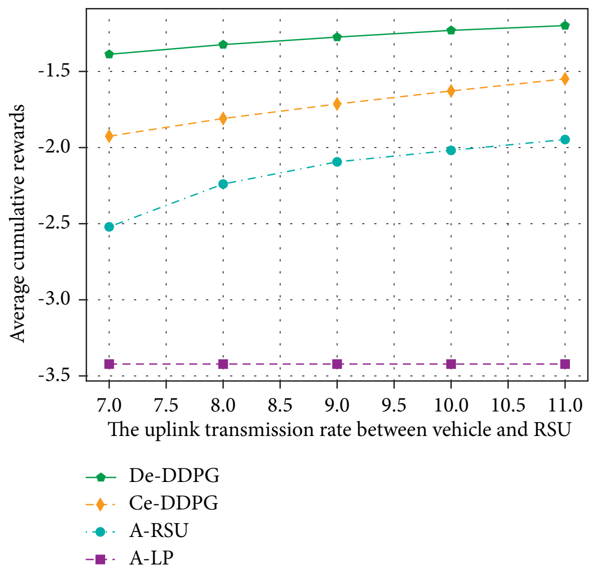


FIGURE 7: Comparison on all algorithms with different uplink transmission rates r_{ij}^{UL}.

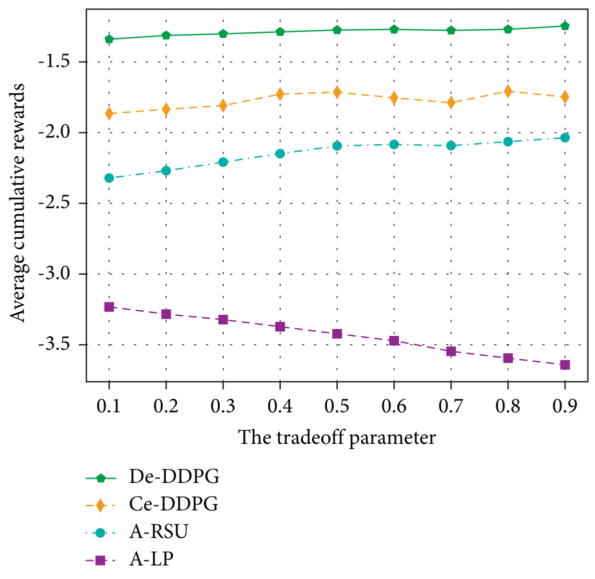


FIGURE 8: Comparison on all algorithms with different trade-off coefficients β.

with the changing of β. As shown in Figure 8, in terms of the average cumulative rewards, the performance of De-DDPG outperforms that of Ce-DDPG, A-RSU, and A-LP no matter the size of β.

6. Conclusions

We propose a computation offloading and resource allocation scheme based on DRL for the MEC-assisted multi-agent with stochastic arrival task model in the IoV environment. To minimize the total weighted cost of the proposed model, we adopt a decentralized multiagent DDPG-based approach (De-DDPG) to solve the nonconvex joint optimization problem. The simulation results demonstrate that our proposed approach has a stable learning capacity and effectively learns the optimal offloading policy and resource allocation to obtain the maximum reward (minimum cost). Compared with the three baseline algorithms, our proposed algorithm has better performance for various parameter configurations. In this paper, the binary offloading decision is used and the task priority is not considered. We will improve these two points in our future work, such as considering partial offloading and task prioritization in this joint optimization problem.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by China National Natural Science Foundation (61572229 and 6171101066).

References

[1] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibanez, “Internet of vehicles: architecture, protocols, and security,” *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2018.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, “A survey on internet of Things: architecture, enabling technologies, security and privacy, and applications,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[3] O. Kaiwartya, A. H. Abdullah, Y. Cao et al., “Internet of vehicles: motivation, layered architecture, network model, challenges, and future aspects,” *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[4] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao, “An edge computing based public vehicle system for smart transportation,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12635–12651, 2020.

[5] K. Zhu, Z. Chen, Y. Peng, and L. Zhang, “Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using LSTM,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4275–4284, 2019.

[6] J. Lin, L. Huang, H. Zhang, X. Yang, and P. Zhao, “A novel lyapunov based dynamic resource allocation for UAVs-assisted edge computing,” *Computer Networks*, vol. 205, no. C, pp. 108710–111286, 2022.

[7] S. Wan, R. Gu, T. Umer, K. Salah, and X. Xu, “Toward offloading internet of vehicles applications in 5G networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4151–4159, 2021.

[8] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, “Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading,” *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[9] Y. Wang, P. Lang, D. Tian et al., “A game-based computation offloading method in vehicular multiaccess edge computing

networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4987–4996, 2020.

[10] J. Zhou, D. Tian, Z. Sheng, X. Duan, and X. Shen, "Distributed task offloading optimization with queueing dynamics in multiagent mobile-edge computing networks," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12311–12328, 2021.

[11] H. Wang, Z. Lin, K. Guo, and T. Lv, "Computation offloading based on game theory in MEC-assisted V2X networks," in *Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, Montreal, QC, Canada, June 2021.

[12] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: a load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[13] G. Wang, F. Xu, and C. Zhao, "QoS-enabled resource allocation algorithm in internet of vehicles with mobile edge computing," *IET Communications*, vol. 14, no. 14, pp. 2326–2333, 2020.

[14] H. Ye, G. Y. Li, and B. H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[15] G. Hong, W. Su, Q. Wen, and P. L. Wu, "RAVEC: an optimal resource allocation mechanism in vehicular MEC systems," *Journal of Information Science and Engineering*, vol. 36, no. 4, pp. 865–878, 2020.

[16] H. Zhang, Z. Liu, S. Hasan, and Y. Xu, "Joint optimization strategy of heterogeneous resources in multi-MEC-server vehicular network," *Wireless Networks*, vol. 28, no. 2, pp. 765–778, 2022.

[17] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Communications*, vol. 17, no. 5, pp. 266–283, 2020.

[18] W. Fan, J. Liu, M. Hua, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5314–5330, 2022.

[19] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.

[20] Z. Ning, Y. Li, P. Dong et al., "When deep reinforcement learning meets 5G-enabled vehicular networks: a distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352–1361, 2020.

[21] S. Xiao, S. Wang, J. Zhuang, T. Wang, and J. Liu, "Research on a task offloading strategy for the internet of vehicles based on reinforcement learning," *Sensors*, vol. 21, no. 18, pp. 6058–6067, 2021.

[22] T. Liu, B. Tian, Y. Ai, L. Li, D. Cao, and F. Y. Wang, "Parallel reinforcement learning: a framework and case study," *IEEE-CAA JOURNAL OF AUTOMATICA SINICA*, vol. 5, no. 4, pp. 827–835, 2018.

[23] X. Xu, B. Shen, S. Ding et al., "Service offloading with deep Q-network for digital twinning empowered internet of vehicles in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1414–1423, 2022.

[24] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763–9773, 2021.

[25] D. Zhang, L. Cao, H. Zhu, T. Zhang, J. Du, and K. Jiang, "Task offloading method of edge computing in internet of vehicles based on deep reinforcement learning," *Cluster Computing*, vol. 25, no. 2, pp. 1175–1187, 2022.

[26] C. Pan, Z. Wang, Z. Zhou, and X. Ren, "Deep reinforcement learning-based URLLC-aware task offloading in collaborative vehicular networks," *China Communications*, vol. 18, no. 7, pp. 134–146, 2021.

[27] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16067–16081, 2020.

[28] Z. Wu and D. Yan, "Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network," *China Communications*, vol. 18, no. 11, pp. 26–41, 2021.

[29] H. Yang, Z. Wei, Z. Feng, X. Chen, Y. Li, and P. Zhang, "Intelligent computation offloading for MEC-based cooperative vehicle infrastructure system: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 7, pp. 7665–7679, 2022.

[30] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[31] N. Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz, "Routing in sparse vehicular ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 8, pp. 1538–1556, 2007.

[32] S. Durrani, X. Zhou, and A. Chandra, "Effect of vehicle mobility on connectivity of vehicular ad hoc networks," in *Proceedings of the 2010 IEEE 72nd Vehicular Technology Conference - Fall*, 2010.

[33] C. Han, M. Dianati, R. Tafazolli, and R. Kernchen, "Throughput analysis of the IEEE 802.11p enhanced distributed channel access function in vehicular environment," in *Proceedings of the 2010 IEEE 72nd Vehicular Technology Conference (VTC 2010-Fall)*, Ottawa, Canada, December 2010.

[34] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[35] Z. Su, Y. Hui, and T. H. Luan, "Distributed task allocation to enable collaborative autonomous driving with network softwarization," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2175–2189, 2018.

[36] K. Zhang, Y. Mao, S. Leng et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[37] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," 2018, https://arxiv.org/abs/1706.02275.

[38] T. P. Lillicrap, J. J. Hunt, A. Pritzel, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," 2019, https://arxiv.org/abs/1509.02971.

Hindawi
Mobile Information Systems
Volume 2022, Article ID 1147819, 9 pages
https://doi.org/10.1155/2022/1147819

Research Article

A Method of Multi-UAV Cooperative Task Assignment Based on Reinforcement Learning

Xiaohu Zhao [1,2] **Hanli Jiang** [1] **Chenyang An** [1] **Ruocheng Wu** [1] **Yijun Guo** [1] **and Daquan Yang** [1]

[1]*School of Information and Telecommunication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*China Academic of Electronics and Information Technology, Beijing 100041, China*

Correspondence should be addressed to Yijun Guo; guoyijun@bupt.edu.cn and Daquan Yang; ydq@bupt.edu.cn

Received 26 April 2022; Accepted 29 June 2022; Published 12 August 2022

Academic Editor: Adarsh Kumar

With the increasing complexity of UAV application scenarios, the performance of a single UAV cannot meet the mission requirements. Many complex tasks need the cooperation of multiple UAVs. How to coordinate UAV resources becomes the key to mission completion. In this paper, a task model including multiple UAVs and unknown obstacles is constructed, and the model is transformed into a Markov decision process (MDP). In addition, considering the influence of strategies among UAVs, a multiagent reinforcement learning algorithm based on SAC algorithm and centralized training and decentralized execution framework, MA-SAC (Multi-Agent Soft Actor-Critic), is proposed to solve the MDP. Simulation results show that the algorithm can effectively deal with the task allocation problem of multiple UAVs in this scenario, and its performance is better than other multiagent reinforcement learning algorithms.

1. Introduction

Unmanned aerial vehicle, also known as UAV, has the characteristics of strong mobility, low safety risk coefficient, no need for personnel to take off, repeatability, and so on. UAV was first used in military fields [1], such as reconnaissance, target strike, air early warning, and electronic jamming. In recent years, UAV technology is developing rapidly, the size of UAV is decreasing, and the cost is getting lower and lower. Therefore, UAV is more and more widely used in civil fields such as sensing [2], cargo transportation, communication relay [3], fire monitoring, and aerial mapping.

With the increasingly complex application scenarios, such as the combination with the Internet of vehicles [4], a single UAV cannot effectively complete complex and diverse tasks. It is important to make multi-UAV perform tasks collaboratively not only to meet the requirement of complicated scenarios but also to make the accomplishment of tasks to cause less time-and-resource consumption.

Task planning is the most important part for the cooperative execution of multi-UAV, and task allocation is the basis of task planning. Task assignment refers to the complex task environment existing in several UAVs; after taking full account of the energy consumption, load, nature, role, and other constraints of UAVs, the coordination between UAVs and various resources is coordinated to assign one or more orderly tasks to each UAV, so as to minimize the time and cost and ensure the efficient and successful completion of tasks to the maximum extent.

The task allocation problem is generally approximated to the path planning problem [5], that is, how to generate a collision-free path from the starting site to the destination to ensure the safety of the vehicle [6]. However, in the multi-UAV environment, not only the collision between UAVs and obstacles but also the collision between UAVs should be considered. At the same time, with the increase of the number of UAVs, the variation of the environment is also increasing. In addition, every action decision of each UAV can be regarded as simultaneous, and no one UAV can know the current decision of other UAVs, so it is more difficult to avoid collisions between UAVs.

Fortunately, reinforcement learning (RL) techniques are emerging to help solve the problem of real-time decision-making in complex and changing environments. The technology allows the drone to learn a strategy to maximize returns or achieve a specific purpose through its constant interaction with the environment.

In this paper, a UAV task allocation model including UAV collision and communication energy consumption is presented; at the same time, an MA-SAC algorithm is proposed to assign tasks and plan paths to UAVs.

The specific works of this paper are as follows:

(i) A multi-UAV task assignment model based on collision and communication energy consumption is proposed

(ii) Based on this assignment model, the dynamic process of task assignment is transformed into MDP

(iii) A multi-agent reinforcement learning algorithm MA-SAC is proposed to solve the MDP process

The rest of this article is organized as follows. Section 2 describes the related work. In Section 3, the multi-UAV task assignment model is presented. Section 4 introduces the task assignment algorithm proposed in this paper. In Section 5, simulation is performed and the results are analyzed. Finally, the works of this paper are summarized in Section 6.

2. Related Work

In the past few years, many researchers have done a lot of research on multi-UAV task allocation model and the algorithm to solve the model. They not only make the model more close to the increasingly complex reality environment but also look for high-performance algorithms. This section will introduce relevant work from these two aspects.

2.1. Task Allocation Model. In various scenarios, different task allocation models need to be established based on a variety of problems that need to be solved by UAV. In the paper [7], and this problem is modeled as a traveling salesman problem (TSP), which minimizes the total flight time and total range of all UAVs by considering the flight capability of UAVs. Jia et al. [8] construct a heterogeneous UAV cooperative multitask allocation scenario by considering kinematic constraints, resource constraints, time constraints, and vehicle path model. Song et al. [9] describe the UAV logistics problem as a mixed integer linear programming problem considering UAV flight time, load, and other constraints. In addition, the task allocation problem of multi-UAV is usually described as multidimensional multiple choice knapsack problem (MMKP) [10, 11], dynamic network flow optimization (DNFO) problem [12], and multiple processors resource allocation (CMTAP) problem [13, 14].

2.2. Task Assignment Algorithm. Task assignment algorithms are mainly divided into optimization algorithm, heuristic algorithm, and reinforcement learning algorithm.

Optimization methods include Hungarian algorithm [15, 16], branch-and-bound method [17], and other commonly used integer linear programming methods. These algorithms are only applicable to scenarios with simple tasks and small UAV scale. Their calculations grow exponentially as the number of UAVs increases, and these algorithms cannot generate an accurate trajectory for UAVs in complex environments. Heuristic algorithms are proposed relative to optimization algorithms, including GA [18], ACO, and PSO that simulate animal behavior in nature. These algorithms are generally combined with other algorithms to solve task assignment problems. In [18], GA is combined with clustering algorithm to solve the task allocation and path planning problems of multiple UAV. In [19], the author proposed two improved heuristic algorithms to solve TSP problems, one is IGA algorithm proposed by improving the coding rules of genetic algorithm, and the other is PSO-ACO algorithm combining PSO and ACO. In [20], the author improves swarm gap algorithm and puts forward three algorithms: location loop (AL), sorting and allocation loop (SAL), and limit and allocation loop (LAL), which solves the task allocation problem of the UAV team in a military operation. However, the heuristic algorithm has the disadvantage of falling into local optimum easily, and the real-time performance of the algorithm is worse and worse with the increase of environment complexity. Therefore, many researchers began to study the application of reinforcement learning in task assignment.

Reinforcement learning is a kind of algorithm that makes an agent learn the optimal strategy through trial and error in the environment. Reinforcement learning has been widely used in UAV mission assignment scenarios over the past few years. In [21], a transaction inspired multiagent reinforcement learning algorithm was proposed to solve the path planning and coordination problems of UAV clusters. In reference [22], the author proposed a MADOL algorithm to enable multiple UAVs to solve the ambiguous BSN allocation problem in an ambiguous boundary scenario. The literature [23] has developed a multiagent reinforcement learning framework, which solves the problem of dynamic resource allocation of UAV communication network in uncertain environment and realizes the balance between performance gain and UAV overhead. In reference [24], the author proposed a multiagent reinforcement learning algorithm, compound-action actor-critic (CA2C), which solves the problem that UAVs perform sensing tasks through cooperative sensing and transmission. In [25], the author proposed an FTA algorithm by combining DQN algorithm with priority experience replay, which effectively solved the problem of UAV task allocation in uncertain environment. In [11], the author proposed a DDQN-per algorithm to solve the task assignment problem of MCS. However, these single-agent algorithms regard the agents in the environment as independent and cannot train a good agent cooperation model. The proposed MADDPG [26] algorithm adopts the method of centralized training and distributed deployment, which well solves the problem of cooperation and competition among multiagent. In [27], the author proposed an MADDPG algorithm, trained the

MADDPG model offline, and then solved the resource allocation problem in the UAV-assisted vehicle network online. However, DDPG algorithm is a deterministic strategy, which may fall into local optimum due to greed. The proposed SAC algorithm [28] introduces entropy, which requires not only maximum reward but also maximum entropy to enhance the spatial exploration ability of agents. Based on the idea of centralized training and separate deployment, this paper applies SAC algorithm to the cooperative task assignment environment of multiple UAVs and proposes an MA-SAC algorithm.

3. Task Assignment Model

Multi-UAV should not only complete each task but also pay attention to their own safety and energy consumption. Figure 1 shows the task allocation framework of multi-UAV. In this paper, the distance from UAV to the mission positions, the collision of UAV, and the communication between UAV and base station are comprehensively considered to establish the task assignment model, and the specific modeling is as follows.

3.1. The Distance between the UAV and the Mission.

This paper considers how to assign multiple UAVs to multiple task points and plan a safe path so as to achieve the goal of reducing the total cost while completing the task quickly and safely. In this paper, the UAV cluster is represented by $V = \{v_1, v_2, v_3, \ldots, v_n\}V = \{v_1, v_2, v_3, \ldots, v_n\}$. The position and track data of each UAV can be obtained by the GPS device carried by the UAV itself, and the data will be transmitted to the MEC layer for calculation. For each UAV $v_i \in V$, (s_{xi}, s_{yi}) is used to represent its current position.

The set of tasks to be completed is represented by $W = \{w_1, w_2, w_3, \ldots, w_n\}$. For each task $w_i \in W$, (s_{wxi}, s_{wyi}) is used to represent task position.

The distance between the UAV v_i and the mission location w_j can be calculated using the following formula:

$$L_{ij} = \sqrt{\left(s_{xi} - s_{wxi}\right)^2 + \left(s_{yj} - s_{wyj}\right)^2}. \tag{1}$$

3.2. UAV Collision.

In order to simulate the real environment, some obstacles are added to the environment to block the route of UAV. At the same time, the collision between UAV and other UAVs is considered. As shown in the picture, there is a certain safety buffer area between the UAV and the obstacles.

The distance between UAVs can be calculated using the following formula:

$$L_{uav} = \sqrt{\left(s_{xi} - s_{xj}\right)^2 + \left(s_{yi} - s_{yj}\right)^2}. \tag{2}$$

Once the distance between UAVs or between UAVs and obstacles is less than the safety zone, UAVs are considered to have a safety risk of collision.

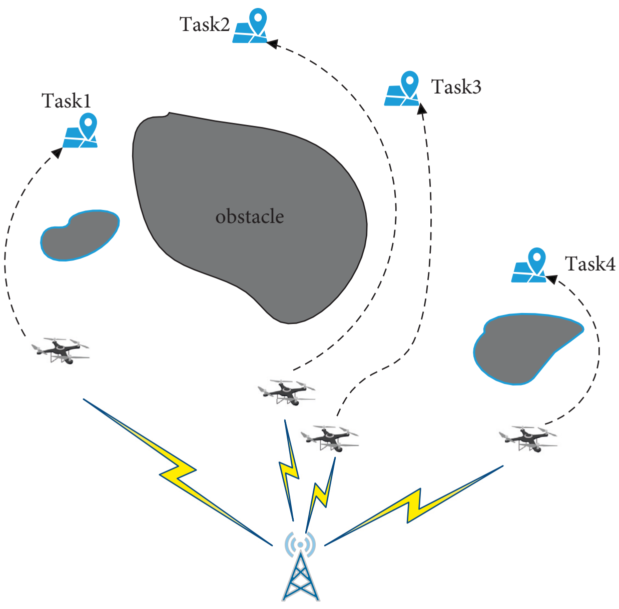


Figure 1: Multi-UAV task assignment model.

3.3. UAV Communication.

In order to grasp the status of UAV in real time, the communication between UAV and base station needs to be considered, and the position of base station is represented by (B_x, B_y). In this paper, UAV's altitude to the ground is h,and the straight-line distance between UAV and base station can be calculated by the following formula:

$$L_{uav-base} = \sqrt{\left(s_{xi} - B_x\right)^2 + \left(s_{yj} - B_y\right)^2 + h^2}. \tag{3}$$

Transmitting the data collected by UAV sensors needs to consume the energy of the sensor node [29]. In order to study the energy loss of UAV transmission, we consider the path loss of UAV communication with base station. In Friis free space model [30], the relationship between signal transmitting power and signal receiving power can be calculated by the following formula:

$$P_R = \frac{P_T G_T G_R \lambda^2}{(4\pi)^2 d^2 \beta}, \tag{4}$$

where P_R is the receiving signal power, P_T is the transmitting signal power, G_T is the transmitting antenna gain, G_R is the receiving antenna gain, λ is the signal wavelength, β is the system loss factor unrelated to propagation, and d is the propagation distance. In this paper, d is the distance between each time slot UAV and the base station $L_{uav-base}$.

In order to ensure normal communication, the power of the attenuated UAV signal needs to be greater than the receiving power of the base station. Therefore, the signal transmitting power of each time slot n of UAV v_i must meet the formula

$$P_{Ti}[n] \geq \frac{(4\pi)^2 d^2 \beta}{G_T G_R \lambda^2} P_{Ri}. \tag{5}$$

The communication energy consumption of each UAV v_i to complete the task can be expressed as

$$E_{\text{com-i}} = \sum_{n\in\mathbb{N}} P_{Ti}[n]\delta, \tag{6}$$

where $\mathbb{N} = (n_1, n_2, n_3, \ldots, n_t)$ is the time slot set for the UAV to complete the task. In this paper, the time slot n is approximated to each step in the simulation. δ is the duration of each time slot n. In this model, δ is set to 1.

The total communication energy consumption of UAV cluster can be calculated by the formula

$$E_{\text{com}} = \sum_{v_i\in V} E_{\text{com}-i}. \tag{7}$$

4. Task Assignment Algorithm

In this section, we consider the application of reinforcement learning in multi-UAV task allocation, apply a soft actor-critic (SAC) algorithm to multiagent environment, and propose an MA-SAC algorithm. This algorithm is usually used to solve the problem described as Markov decision process (MDP). So, this section will introduce the MDP of this model, SAC algorithm and MA-SAC algorithm in turn.

4.1. Markov Decision Process. MDP is usually composed of state, action, and reward function. Therefore, the MDP of the model can be described as follows.

4.1.1. State. In this process, the state space is composed of the position and speed of the UAV, the distance between the UAV and the destination, and the collision risk of the UAV.

4.1.2. Action. The action space is usually the optional action set of all UAVs in different states. In this model, the action space of UAV is expressed as < front, back, left, right, hover >.

4.1.3. Reward. In this model, when multiple UAVs are faced with multiple tasks, this paper aims to reasonably allocate task targets and carry out path planning for each UAV, so that each task can be completed safely and quickly with the minimum total energy consumption. Therefore, for UAV v_i, the reward can be described as

$$R_i = R_F + R_L + R_c - E_{\text{com}-i}. \tag{8}$$

The task assignment problem can be described as

$$\max \sum_{v_i\in V} R_i, \tag{9}$$

$$\bigcup_{i=1}^{n} w_{ij} = V, \tag{10}$$

$$\bigcup_{i=1}^{n} v_{ij} = W, \tag{11}$$

where R_F is the reward for completing the task, and the value is constant. R_c is the collision reward. R_L is the distance reward. In order to guide the UAV to the mission point, it can be expressed as $R_L = -\min L_{ij}$, $j \in (1, 2, \ldots, n)$, w_{ij} indicates that the mission w_i is carried out by UAV v_j, and v_{ij} indicates that UAV v_i performs mission w_j. Formula (10) means that only one UAV can be assigned to perform each task, and formula (11) means that each UAV can only perform one task.

4.2. SAC Algorithm. SAC algorithm is a kind of off-policy reinforcement learning algorithm. This paper is improved based on SAC algorithm proposed in [31]. The algorithm improves the critical network on the first version of SAC algorithm [32]. It removes the value network and uses two Q networks. Therefore, the SAC algorithm has one actor network, two critic networks, and two target-critic networks. Among them, the actor network is used to give the corresponding action according to the change of state, and the critic network is used to calculate the Q value to evaluate the action. In order to solve the overestimation problem, the SAC algorithm adopts a pair of independent critic network and takes the smaller value of the two when updating. In order to stabilize the training of Q network, the SAC algorithm introduces a pair of target-critic networks whose update frequency is less than the critic network.

In order to prevent the strategy from getting into trouble due to greed, it is necessary to increase the random exploration ability of the algorithm, so SAC introduces entropy regularization. When the strategy distribution is more uniform, the entropy of the strategy is greater, and the random exploration ability of the algorithm is stronger. Therefore, the objective function of SAC algorithm not only requires the maximum final reward but also the maximum entropy. Its objective function can be expressed as

$$\begin{aligned} J(\pi) &= \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t) + \alpha H(\pi(\cdot\mid s_t))\right] \\ \pi^*_{\max} &= \operatorname{argmax}_\pi \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t) + \alpha H(\pi(\cdot\mid s_t))\right], \end{aligned} \tag{12}$$

where $H(\pi(\cdot\mid s_t))$ is the entropy of strategy, $r(s_t,a_t)$ is the reward for time t, and $\pi^*_{\max}$ is the optimal strategy.

4.3. MA-SAC Algorithm. Figure 2 shows the MA-SAC algorithm that we proposed by improving SAC algorithm based on the multi-UAV task allocation model. MA-SAC algorithm is based on actor-critic network framework. In this multi-UAV environment, each UAV has an actor network, a target-actor network, two critic networks, and two target-critic networks, which are all composed of fully connected neural networks.

In the multi-UAV environment, UAV itself is not only an intelligent body but also a part of the environment of other UAVs. Therefore, for the critic network of each UAV, we not only input the environmental state into the critic network. The actions of other UAVs are also fed into the critic network to calculate the Q by a part of the overall environment. SAC, like DDPG and other algorithms, introduces the experience replay mechanism to reduce the

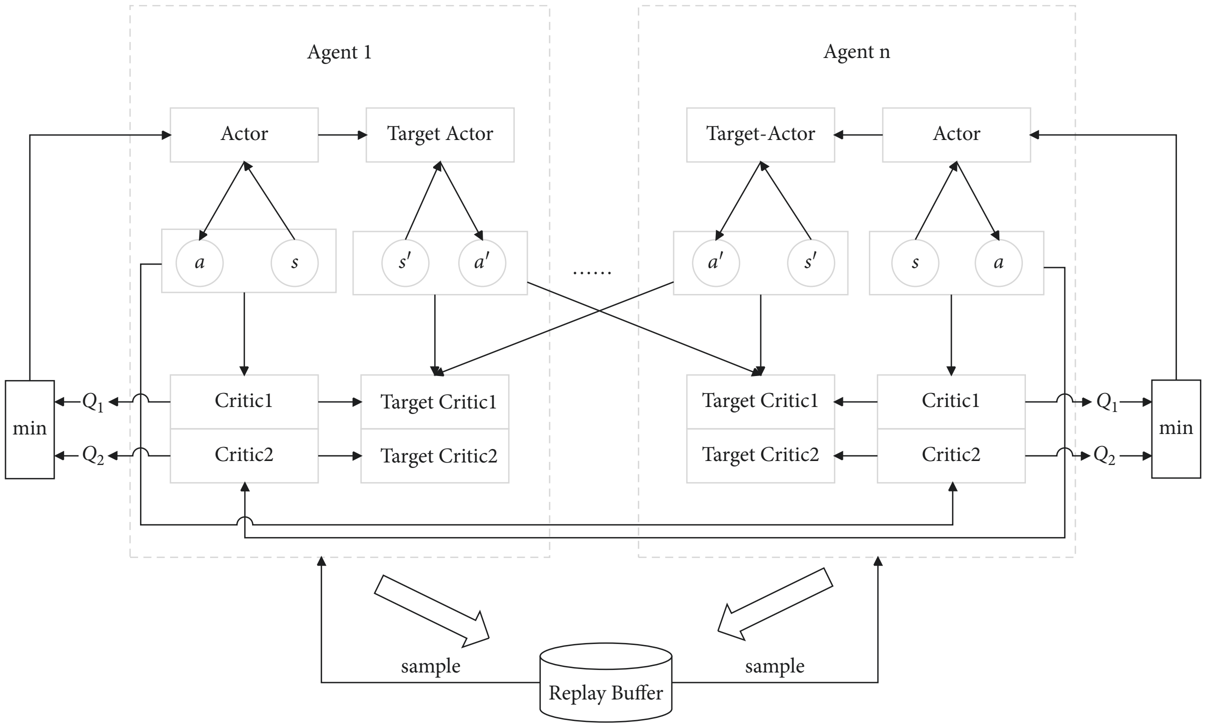


Figure 2: Actor and critic neural network of MA-SAC.

correlation between data. Therefore, the whole training process is divided into two parts: experience collection and network training. In the experience gathering phase, the agent performs the actions generated in each step, and then stores the tuples that include states, action, next state, and reward $\langle S, A, S\prime, R\rangle$ into the replay buffer.

When the data in the replay buffer reaches the threshold, the network training stage can be entered. At each step, some data will be sampled from the replay buffer to update the parameters of actor networks and critic networks. The actor network is trained by the strategy gradient. For each UAV $v_i \in V$, the actor network update targets are as follows:

$$J(\theta_i) = \mathbb{E}_{X,a\sim D}\left[\alpha \log(\pi_i(a_i|s_i)) - Q_i^{\pi}(X, a_1, \ldots, a_n)|_{a_i=\pi_i(s_i)}\right], \tag{13}$$

where π_i represents the policy network of the agent i, $\theta_i \in \{\theta_1, \theta_2, \ldots, \theta_n\}$ represents the parameter of the policy network π_i, and X represents the current status of all agents.

Critic networks are updated by minimizing the loss function as a goal. The loss function is the mean square error that can be calculated by the formula:

$$\mathscr{L} = \mathbb{E}_{(X,a,r,X')\sim D}\left[\left(Q_i^{\pi}(X, a_1, \ldots, a_n) - y_i\right)^2\right], \tag{14}$$

$$y_i = r_i + \gamma\mathbb{E}\left[Q_i^{\pi}(X', a_1', \ldots, a_n')|_{a_i'=\pi_{\overline{\theta}_i}(s_i')} - \alpha \log\left(\pi_{\overline{\theta}_i}(a_i'|s_i')\right)\right], \tag{15}$$

where X' represents the next status of all agents, a_i' represents the next action of the agent i, and s_i' represents the next state of the agent i.

To ensure the stability of training, the parameters of actor networks and critic networks will be copied to the corresponding target networks in each iteration. Here, the algorithm adopts the soft update method, so in each step, some actor and critic network parameters are updated to the corresponding target network, which can be calculated by the formula

$$\overline{\psi} \longleftarrow \tau\psi + (1-\tau)\overline{\psi}, \tag{16}$$

$$\overline{\theta} \longleftarrow \tau\theta + (1-\tau)\overline{\theta}, \tag{17}$$

where $\overline{\psi}$ is the parameter of target-critic network, ψ is the parameter of the critic network, and τ is the update ratio.

The pseudocode of the MA-SAC algorithm is demonstrated in Algorithm 1, and the meanings of the parameters are shown in Table 1.

5. Experimental Results and Analysis

In this section, the performance of MA-SAC algorithm in multi-UAV task assignment environment is studied. We use the Pytorch deep learning framework to simulate this scenario and compare it with MADDPG algorithm. Table 2 shows the relevant hyperparameters of the algorithm simulation in this paper.

In this experiment, we constructed an environment in which multi-UAV cooperate to complete tasks. The environment consists of three UAVs, three mission positions, one obstacle, and a base station to communicate with the UAVs. Firstly, the MADDPG algorithm proposed in reference [26] is selected to compare the convergence performance. Figure 3 shows the convergence process of MA-

Table 1: Explanation of variables and functions in the algorithm of MA-SAC.

Variable	Explanation
episodes	The maximum number of iterations
steps	The maximum step length for each iteration
D_{size}	The amount of data in the replay buffer
B_{size}	Sampling number

```
(1) Initialize environment
(2) Initialize critic network and actor network
(3) Initialize max episodes, replay buffer, batch size
(4) for $episode \in [1, episodes]$ do
(5)     Reset environment
(6)     Get current state $s_i$ for each agent, $i$
(7)     for $step \in [1, steps]$ do
(8)         Select actions $a_i$ for each agent $v_i$
(9)         Get all agents next states $s_i'$ and rewards $r_i$
(10)        Store $< a_i, s_i, s_i', r_i >$ to replay buffer $D$
(11)        if $D_{\text{size}} > B_{\text{size}}$ then
(12)            Sample batch $B$ from replay buffer $D$
(13)            for $v_i$, where $i = 1:N$ do
(14)            Update the critic network
(15)            Update the actor network
(16)            Update the target network according to formulas (15), (16)
(17)            end for
(18)        end if
(19)    end for
(20) end for
```

Algorithm 1: Algorithm of MA-SAC.

Table 2: The parameters of simulation.

Parameter	Value
Number of UAVs	3
Number of tasks	3
Number of obstacles	1
Number of base stations	1
Steps of episode	35
Capacity of replay buffer	1000000
Number of network neurons	128
Learning rate	0.001
Discount factor of reward	0.99
Update ratio of target network τ	0.001

SAC algorithm and MADDPG algorithm during training in this environment. In this experiment, we performed 50,000 training episodes and averaged the rewards every 1,000 episodes. By comparing the two algorithms, it can be found that the proposed MA-SAC algorithm can finally converge to around 300, while the MADDPG algorithm finally converges to around 220. It can be seen that the convergence speed of the two algorithms is similar in this scenario, but the convergence result of the MA-SAC algorithm is better than that of the MADDPG algorithm, because the training goal of the MA-SAC algorithm is not only to maximize the reward of the drone but also to maximize the entropy of the UAV

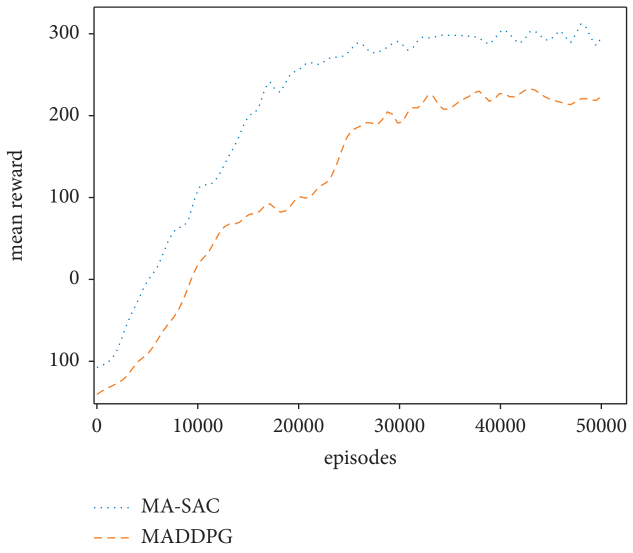


Figure 3: Reward of different algorithms.

strategy. This increases the ability of the UAV to explore the space, thereby improving the performance of the algorithm.

To verify the effectiveness of the algorithm in this scenario, we conducted 500 episodes of tests on the MA-SAC algorithm in this environment and compared it with other

Table 3: Task completion rate.

Algorithm	Task completion rate (%)
MA-SAC	95.16
MADDPG	92.76
COMA	82.67
VDN	68.34

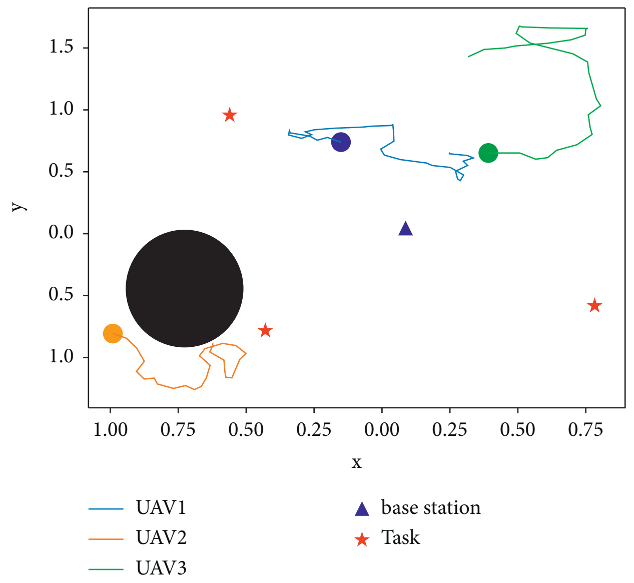


Figure 4: Rendering of task assignment during 0 w episodes of training.

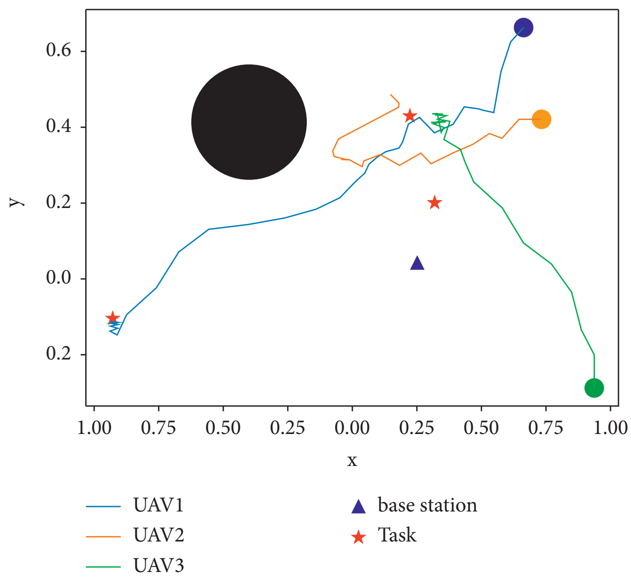


Figure 5: Rendering of task assignment during 2 w episodes of training.

multiagent reinforcement learning algorithms. As shown in Table 3, the task completion rate of the MA-SAC algorithm reaches 95.16%, which is a great improvement compared with that of the COMA and VDN algorithms, and the task

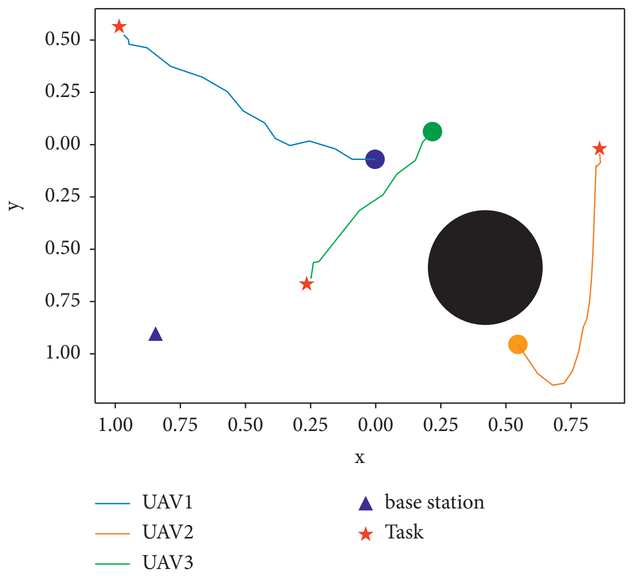


Figure 6: Rendering of task assignment during 5 w episodes of training.

completion rate is also increased by 2.4% compared with the MADDPG algorithm.

Figure 4 shows the dynamic assignment process of UAVs in the task area before training. At this time, none of the three UAVs has learned any strategy, so they are in an exploration state in the environment. It can be seen from the route of the UAV in the task assignment process that the UAV does not have a clear mission target at this time, and they move randomly in space. UAV 2 even collides with obstacles.

Figure 5 shows the rendering of the multi-UAV task assignment process when using the proposed MA-SAC algorithm for 20,000 episodes of training. It can be seen that although the UAVs have learned to approach the mission point at this time, there is no coordination between them. Both UAV 2 and UAV 3 flew to the same mission location, resulting in not all missions being completed.

Figure 6 shows the effect of the task assignment process of the UAV when the training reaches 50,000 episodes. At this point, the trained model can already solve the task assignment problem in this environment well. UAVs not only consider their distance when assigning tasks but also take into account the strategies of other UAVs and cooperate with each other to complete all tasks in the mission area. At the same time, UAVs have also learned to stay away from obstacles to reduce their own risks when completing tasks. It can be seen that UAV 2 is relatively close to the obstacle at the beginning, so there is a possibility of collision. In order to ensure its own safety, it first flies away from the obstacle, and then flies to the mission location after reaching the safe area.

6. Conclusions

In this paper, a multi-UAV cooperative task assignment model in complex environment is constructed by considering UAV distance, collision, and communication. Meanwhile, we propose an MA-SAC algorithm to solve the model by combining the SAC algorithm of deep reinforcement learning with multiagent framework of centralized training and decentralized execution. Simulation results show that the MA-SAC algorithm is superior to the MADDPG algorithm in convergence result in multi-UAV task allocation environment. In terms of task completion rate, the model trained by the MA-SAC algorithm also achieved a better result.

In the future work, more complex factors will be considered in the environment, such as making the communication model more suitable for real scenes and weather changes. At the same time, it will also study the larger-scale dynamic task allocation of UAV. Since this paper only studies the UAV cooperation scenario, the UAV task allocation in the countermeasure scenario will be studied in the future.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation of China (11974058 and 61901050); Beijing Nova Program (Z201100006820125) from Beijing Municipal Science and Technology Commission; Beijing Natural Science Foundation (Z210004); and State Key Laboratory of Information Photonics and Optical Communications (IPOC2021ZT01), BUPT, China.

References

[1] L. Bertuccelli, H. L. Choi, and P. Cho, "Real-time multi-UAV task assignment in dynamic and uncertain environments," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, p. 5776, Ontario, Canada, September, 2009.

[2] S. Huang, A. Liu, S. Zhang, N. N. Wang, and N. N. Xiong, "BD-VTE: a novel baseline data based verifiable trust evaluation scheme for smart network systems," *IEEE transactions on network science and engineering*, vol. 8, no. 3, pp. 2087–2105, 2021.

[3] K. Zhu, X. Xu, and Z. Huang, "Energy-efficient routing algorithms for UAV-assisted mMTC networks," in *Proceedings of the 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, IEEE, Istanbul, Turkey, September, 2019.

[4] K. Gao, F. Han, P. Dong, N. Xiong, and R Du, "Connected vehicle as a mobile sensor for real time queue length at signalized intersections," *Sensors*, vol. 19, no. 9, p. 2059, 2019.

[5] X. Tao and A. S. Hafid, "Trajectory design in UAV-aided mobile crowdsensing: a deep reinforcement learning approach," in *Pocedings of the IEEE ICC*, June, 2021.

[6] H. Li, J. Liu, K. Wu, Z. Yang, R. W. Liu, and N. Xiong, "Spatio-temporal vessel trajectory clustering based on data mapping and density," *IEEE Access*, vol. 6, Article ID 58939, 2018.

[7] N. Ozalp, U. Oztop, and E. Oztop, "Cooperative multi-task assignment for heterogonous UAVs," in *Proceedings of the 2015 International Conference on Advanced Robotics (ICAR)*, pp. 599–604, (ICAR), Istanbul, Turkey, July, 2015.

[8] Z. Jia, J. Yu, X. Ai, X. Xu, and D. Yang, "Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm," *Aerospace Science and Technology*, vol. 76, pp. 112–125, 2018.

[9] B. D. Song, K. Park, and J. Kim, "Persistent UAV delivery logistics: MILP formulation and efficient heuristic," *Computers & Industrial Engineering*, vol. 120, pp. 418–428, 2018.

[10] L. R. Rodrigues, J. P. P. Gomes, and J. F. L. Alcântara, "Embedding remaining useful life predictions into a modified receding horizon task assignment algorithm to solve task allocation problems," *Journal of Intelligent and Robotic Systems*, vol. 90, no. 1-2, pp. 133–145, 2018.

[11] M. Alighanbari and J. How, "Robust decentralized task assignment for cooperative UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 6454, San Francisco, California, August, 2006.

[12] K. E. Nygard, P. R. Chandler, and M. Pachter, "Dynamic network flow optimization models for air vehicle resource allocation," in *Proceedings of the American Control Conference*, pp. 1853–1858, Arlington, VA, USA, June, 2001.

[13] S. Fei, C. Yan, and S. Lin-Cheng, "UAV cooperative multi-task assignment based on ant colony algorithm," *Acta Aeronautica et Astronautica Sinica*, vol. 29, no. 5, pp. 188–s189, 2008.

[14] E. Edison and T. Shima, "Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms," *Computers & Operations Research*, vol. 38, no. 1, pp. 340–356, 2011.

[15] K. Kim and C. S. Hong, "Optimal task-UAV-edge matching for computation offloading in UAV assisted mobile edge computing," in *Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–4, IEEE, Matsue, Japan, September, 2019.

[16] B. S. MirzaeiniaA and M. Hassanalian, "Drone-station matching in smart cities through Hungarian algorithm: power minimization and management," in *Proceedings of the AIAA Propulsion and Energy 2019 Forum*, p. 4151, Indianapolis, August, 2019.

[17] S. J. Rasmussen and T. Shima, "Branch and bound tree search for assigning cooperating UAVs to multiple tasks," in *Proceedings of the 2006 American Control Conference*, p. 6, June, 2006.

[18] Y. Ma, H. Zhang, Y. Zhang, R. Gao, Z. Yang, and J. Yang, "Coordinated optimization algorithm combining GA with cluster for multi-UAVs to multi-tasks task assignment and path planning," in *Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pp. 1026–1031, Edinburgh, UK, July, 2019.

[19] J. Chen, F. Li, and Y. Li, "Travelling salesman problem for UAV path planning with two parallel optimization algorithms," in *Proceedings of the 2017 Progress in Electromagnetics Research Symposium - Fall (PIERS - FALL)*, pp. 832–837, Singapore, November, 2017.

[20] J. Schwarzrock, I. Zacarias, A. L. C. Bazzan, R. Q. de Araujo Fernandes, L. H. Moreira, and E. P. de Freitas, "Solving task allocation problem in multi Unmanned Aerial Vehicles systems using Swarm intelligence," *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 10–20, 2018.

[21] A. A. Khalil, A. J. Byrne, and M. A. Rahman, "Efficient uav trajectory-planning using economic reinforcement learning," 2021, https://arxiv.org/abs/2103.02676.

[22] Y. Zhang, Z. Mou, F. Gao, L. Xing, J. Jiang, and Z. Han, "Hierarchical deep reinforcement learning for backscattering data collection with multiple UAVs," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3786–3800, 2021.

[23] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, 2020.

[24] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative Internet of UAVs: distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6807–6821, 2020.

[25] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerospace Science and Technology*, vol. 92, pp. 588–594, 2019.

[26] H. Qie, D. Shi, T. Shen, X. Xu, Y. Wang, and L. Wang, "Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, Article ID 146264, 2019.

[27] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 131–141, 2021.

[28] R. Lowe, Y. Wu, and A. Tamar, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2017, https://arxiv.org/abs/1706.02275.

[29] M. Wu, L. Tan, and N. Xiong, "A structure fidelity approach for big data collection in wireless sensor networks," *Sensors*, vol. 15, no. 1, pp. 248–273, 2014.

[30] T. S. Rappaport, *Wireless communications -- principles and practice*, Prentice Hall PTR, Hoboken, New Jersey, 2002.

[31] T. Haarnoja, A. Zhou, and K. Hartikainen, "Soft actor-critic algorithms and applications," 2018, https://arxiv.org/abs/1812.05905.

[32] T. Haarnoja, A. Zhou, and P. Abbeel, "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the International Conference on Machine Learning*, pp. 1861–1870, PMLR, Chengdu, China, July, 2018.