Task Migration Strategy in Vehicular Networks Based on Reinforcement Learning

Zou Jing, Gong Qishuai, Wang Zhe
Guangxi Minzu University School of Artificial Intelligence, Nanning, China

Abstract: With the research and application of edge computing in vehicular networks, computing tasks can be offloaded from vehicles to roadside edge servers to reduce system service latency. However, as vehicles move, the computing tasks need to be migrated from one edge server to another. Predicting the vehicles movement trajectory and formulating a reasonable task migration plan for this is a key challenge that needs to be addressed. Traditional computing offloading methods cannot be directly applied in vehicular networks. Therefore, this paper constructs a vehicular task offloading system based on a multi-layered computing network, introduces a Markov mobility model to describe the vehicle movement trajectory, and solves the optimal migration path problem. Since this problem is NP-hard, a solution method based on a constrained Markov model is proposed, along with an Actor-Network Primal-Dual Deep Deterministic Policy Gradient (ANPD-DDPG) algorithm based on reinforcement learning to achieve the optimal solution. Finally, in simulation experiments, the proposed method is compared with existing research, showing about a 33% reduction in system delay and migration cost. The characteristics of the ANPD-DDPG algorithm in terms of convergence speed and system delay are also analyze

Keywords: Internet of Vehicles; Edge Computing; Task Offloading; Task Migration

1. Introduction

In recent years, with the progress of on-board computing power and high-speed communication technology, the intelligent degree of vehicles has been rapidly improved, accelerating the construction and development of the Internet of Vehicles (Internet of V ehicle, IoV)[1]. Compared to the local vehicle computing, IoV needs to deal with a large number of computing-intensive and delay-sensitive tasks[2], This forces traditional central cloud computing to move mobile edge computing (Mobile Edge Computing, MEC), providing vehicles with more resources than local services in edge devices closer to the road[3-4]。Thus, the service quality of the IoV (Quality of Service, QoS)[5] As the core element of vehicle driving decision, we need to consider the vehicle task unloading and task migration with vehicle movement.

In terms of task unloading, literature [6] for mobile computing unloading review, analyzes the traditional heuristic calculation unloading strategy and online learning calculation unloading strategy their advantages and applicable scenarios, and points out that due to the uncertainty of the vehicle moving position and speed may lead to traditional unloading strategy failure, need to consider the dynamic change of the vehicle and environment. Based on this, literature [7] simulates the mobile task unloading as a Markov decision (Markov Decision Process, MDP) process, designed the task unloading strategy according to the edge server state, vehicle movement trajectory and task buffer queue, and minimized the system delay under the bandwidth constraint, and uses the one-dimensional search algorithm to solve the optimal unloading decision. Similarly, literature [8] in the energy collection MEC system research unloading problem, the system delay time and task failure cost as the optimization target, proposed a dynamic online unloading algorithm based on Lyapunov optimization, the algorithm is low load and in the unloading of CPU weight, vehicle position and energy loss during transmission, need not consider the task type, wireless channel status and energy collection process information, improve the online performance and robustness of the algorithm. Literature [9] considers the MEC system unloading problem from the

perspective of resource allocation to minimize the system delay constraintEnergy consumption is the goal, by proving that there are threshold characteristics for the optimal solution, the unloading priority of the vehicle, the task whose priority is above the threshold is fully unloaded, and the task below the threshold is partially unloaded. Also with energy consumption as the optimization goal, literature [10] jointly consider the comprehensive energy consumption of vehicle and edge server, the server selection, bandwidth resource allocation, vehicle trajectory prediction and computational resource allocation joint optimization, put forward the specific application constraints of mixed nonlinear planning problem (Mixed Integer Nonlinear Program, MINLP) and design the greedy heuristic algorithm solution. Based on the mobility and heterogeneity of vehicles, literature [11] proposed the calculation and unloading scheme of MEC system under the Internet of Vehicles, designed the self-sufficient management framework of deep reinforcement learning, and solved it by establishing MDP model and deep reinforcement learning algorithm. It should be pointed out that the environmental data involved in the above studies are instantaneous, and continuous environmental changes may lead to the increase of the error in the strategy, and then lead to the failure of the strategy or the increase of algorithm complexity, affecting the QoS requirements of the Internet of Vehicles

On the other hand, high-speed moving vehicles make the unloading task need to migrate between different servers, and the ideal migration target should be dynamically adjusted with the driving trajectory of the vehicle. Based on vehicle trajectory history data, literature [12] proposes a deep reinforcement network algorithm to achieve the best task migration strategy, and also illustrates that the data-driven and machine learning methods are more suitable for the design of dynamic task transfer strategy. Accordingly, literature [13] considers the offline scenario, proposes a server coordination algorithm based on dynamic planning to realize the joint optimization of delay and migration cost in MEC system, models the prediction process of vehicle trajectories as MDP by proving the optimality of the solution, and designs the reinforcement learning algorithm to realize the optimal transfer strategy. Different from the above studies, literature [14] considers the data fault tolerance, simulates the task migration process as a partially observable MDP, takes the vehicle as the decision center to realize the online migration strategy design through trajectory prediction, and proposes an offline strategy actor-critic algorithm, Off-Policy Actor Critic (OPAC), to achieve better QoS performance than the above studies. On the other hand, literature [16] addresses the uncertainty of trajectory prediction, using Lyapunov method to convert vehicle movement into real-time control problem, proposed edge service performance optimization under the constraint of long-term migration cost budget, and designedMarkov Approximation (Markov Approximation, MA) algorithm for real-time decision making. Thus, data-driven machine learning method can make up for the shortcomings of instantaneous information and can improve system strategy real-time, but the vehicle and service data interaction during the mobile network pressure should not be ignored, such as the literature [12] strategy while real-time prediction, but failed to consider the car network bandwidth and delay pressure, literature [13]-[16] research is ignoring the edge of the server and system computing resources limited, unable to cope with the car networking multi-task dynamic migration.

In view of the above deficiencies, this paper proposes the task unloading and migration strategy based on reinforcement learning, jointly considers the task unloading ratio and edge resource constraint to realize adaptive unloading, and establishes a Markov mobile model to predict the vehicle trajectory to realize task migration.first, Build a multi-layer computing framework with vehicles, roadside edge servers and central cloud servers as systems, Establish the Internet of Vehicles system model; then, Under the joint constraints of the system computing resources and the time delay, Raise the time delay and cost minimization problem of the system task unloading and migration process, Through the constrained Markov decision process (Constrained Markov Decision Process, CMDP) solves the optimal path; Then, the original-dual depth deterministic strategy gradient of the actor network based on reinforcement learning was designed (Actor-Network Primal-Dual Deep Deterministic Policy Gradient, ANPD-DDPG) algorithm to realize the system strategy; last, The advantages of the proposed

method and the designed algorithm in reducing the system cost and improving the performance gain are verified.

System model and problem description

system model

The multi-layer computing framework with vehicle, roadside edge server (Roadside Edge Server, RES) and central cloud server as the system is shown in Figure 1. Suppose that the number of edge servers in the system is in s, $s \in \{1,2,\ldots,S\}$ and the number of vehicles is in u. $u \in \{1,2,\ldots,U\}$ The vehicle can select the local calculation or unload the task to the RES based on its own task situation, and assumes that the system is running in a fixed cycle and the RES does not switch between different tasks in the time step of a certain cycle. Represents the movement in the city of the vehicle at different times, while ensuring that the server can migrate the next task only after the current task migration is completed. As set in reference [17], all vehicles entered the system at time t=1 and exit at time t= T.

To describe the movement characteristics of the vehicle, a Markov mobility model is introduced[18], Whether the vehicle u is $i_{u,s}^t$ located in the server s at time t is represented as a binary indicator variable. $i_{u,s}^t = p\left[i_{u,s}^t\right]$ If the vehicle mobility is not fully determined in the predicted time step, take the variable in order to indicate the probability that the vehicle u is at the server s at time t; $p\left[i_{u,s}^t\right]$ therefore, the value is a continuous variable in [0,1].

Task Uninstall: Use to indicate the vehicle u unload task by the server. S is calculation, because the task may need to be migrated, all the servers unloaded by the task under calculation are not necessarily the servers where the vehicle u is located. Also ensure that the vehicle u will always maintain communication for the server s that it serves. The position of the vehicle u is represented by the $q_{u,s}^t$ variables and is $i_{u,s}^t$ given by the $p\left[i_{u,s}^t\right]$ predictions.

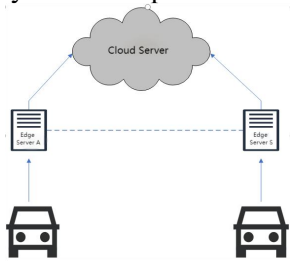


Figure 1. Multi-Level Calculation and Unloading System Network

Fig.1 Multi-level Computational offloading network

Task Migration Process: Figure 2 shows the flow chart of the system task migration. The process begins with the vehicle unloading a task at the selected RES. Coordinate multiple RESs in cloud services. By communicating with RES, cloud services collect the servers current available resources and mobile patterns of the vehicle through resource tracker and mobile data collector. The movement predictor uses the movement data of the vehicle to predict the movement trajectory of the vehicle. This data is then sent to the Migration Plan generator to generate the migration plan. The Task Migration Plan Generator subsequently broadcasts the generated migration plan to all the RESs.

Migration Plan generation: Figure 3 shows the migration process of an unloading task. It contains a start and end node that sets the server index of both the start and end nodes to s= 0. Define other nodes in the migration graph as server-time pairs, representing where tasks may migrate in cycle T. At the vertex (s1,t1) and (s2,t2) between the edges, where, represents the slave server s $s_1 \neq s_2$ 1 to the servers sThe ation of 2, which begins at time t1, and ends at time t2. During the migration, the task remains on the source server while building a copy of the task on the target server. If, the edge represents simply stays on the server s $s_1 = s_2$ 1. Associate each edge with a weight that represents the migration cost of adopting that path on the migration graph, as shown in the solid line in Figure 3. Figure 3 depicts a migration graph that underwent three time steps in two server systems. For the task of vehicle u, any feasible migration path from the start node to the end node in Fig. And tasks can only be migrated to one server within any time step t. So, define an indicator indicating if from server s h_{s_1,s_2}^{u,t_1,t_2} 1 To sThe path of 2 and that taken from time t1 To tThe path of 2 is included in th h_{s_1,s_2}^{u,t_1,t_2} e migration plan, then the indicator =1, for example, the dashed migration plan in Figure 3, and =0 if not included in the migration h_{s_1,s_2}^{u,t_1,t_2} plan. The indicator is used to ensure that the selected migration path of the migration plan is present in the h_{s_1,s_2}^{u,t_1,t_2} migration map to ensure that (i) a

node on the migration map, and $t_1 \in (|T|<t)$ (ii) the start of one migration plan, just at the end of another migration occurs. Where represents the set of all time steps less $\{1,2,\ldots,T\}$ than t, for a similar definition. In addition, $t_2 \in (|T+1|>t)$ representing all collectio $s \in \{0,|S|\}$ ns of servers, to ensure that all migration schedules $\{1,2,\ldots,S\}$ start at time t=1 and end at time t=T, is defined as follows:

$$\sum_{t_1 \in (|T|<t)} \sum_{s_2 \in \{0,|S|\}} h_{s_1,s}^{u,t_1,t_2} = \sum_{t2 \in (|T+1|>t)} \sum_{s_2 \in \{0,|S|\}} h_{s,s_2}^{u,t_1,t_2}$$
$$\sum_{s \in |S|} h_{0,s_2}^{u,0,1} = \sum_{s \in |S|} h_{0,s_2}^{u,T,T+1} = 1 \quad (1)$$

To define. By ensuring that if vehicle u the task is going from server s h_{s_1,s_2}^{u,t_1,t_2} $q_{u,s}^{t}$ $q_{u,s}^{t}$ 1 Migrated to the servers s2, then it will be given by the s1 provides services until the migration is complete, as $q_{u,s}^{t}$ defined below

$$q_{u,s}^{t} = \sum_{S_1 \in (|S|)} \sum_{t_1 \in \{|T|\}} \sum_{t_2 \in (|T| \geq t)} h_{s_1,s}^{u,t_1,t_2} - \sum_{S_2 \in (|S|)} \sum_{t_3 \in \{|T|\}} \sum_{t_4 \in (|T| \geq t)} h_{s,s_2}^{u,t_1,t_2} \quad (2)$$

Definition, indicating that the task of vehicle u at time t is migrated to server s. It is $g_{u,s}^{t}$ defined as follows

$$g_{u,s}^{t} = \sum_{s_1 \in (|S|)} \sum_{t_1 \in \{|T|\}} \sum_{t_2 \in (|T| \geq t)} h_{s_1,s}^{u,t_1,t_2} \quad (3)$$

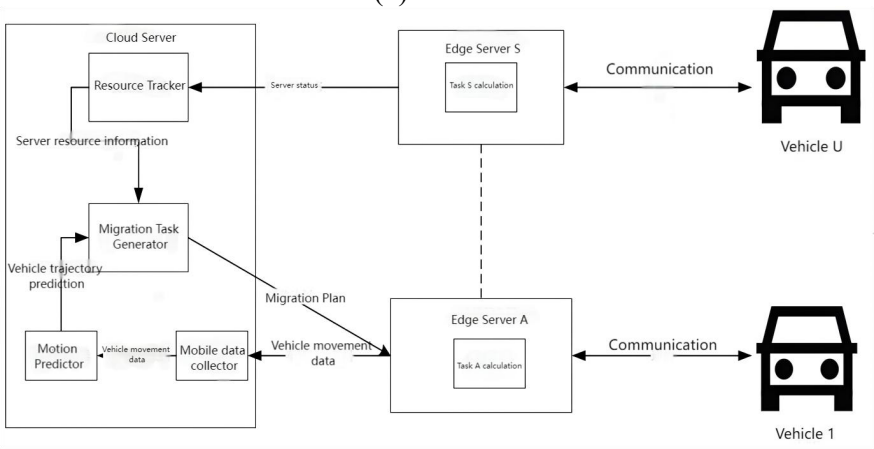


Figure 2. System Model

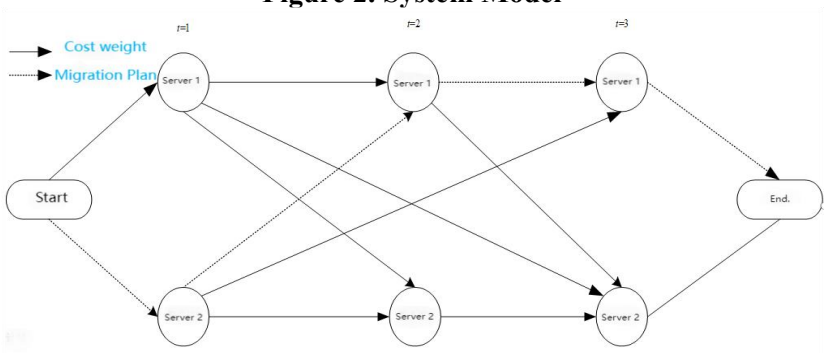


Figure 3. Task Migration Communication Model

The entire communication cycle T is divided into t time steps, and the vehicle moves at a certain speed. Some of these tasks are unloaded to the RES. Considering the actual situation, the limitation of computing power makes the vehicle unable to fully undertake the local task calculation, so the vehicle must unload part of the task to the RES. Within each time step t, the vehicle u selects to communicate with the server s. Suppose that the transmission rate of the network can meet the task unloading requirements. Therefore, the transmission rate from the vehicle u to s is as follows:

$$R_{s,u} = C \log_2(1 + r_{s,u}) \quad (4)$$

Where C represents the communication bandwidth of the system and the signal to-noise ratio from u to s, and its formula can be expressed as: $r_{s,u}$

$$r_{s,u} = \frac{p_u g_{s,u}(t)}{\sum_{n \neq u} p_u g_{s,n}(t) + b_u(t) P_{NLOS} + \sigma_s^2} \quad (5)$$

It represents the additive Gaussian white noise (Additive White Gaussian Noise, AWGN) received by s, whose distribution is, and Pu is the transmission power of u, indicating the presence of obstacles from u to s σ_s^2 $(0,\sigma_s^2)$ $b_u(t)=\{0,1\}$ [19], the non-horizon

transmission loss between u and s (Non-Line-of-Sight Transmission Loss, NLOSTL). It represents the c $g_{s,u}(t)$ hannel gain from u to s at the time step t, e P_{NLOS} xpressed as follows:

$$g_{s,u}(t) = \alpha_0 d_u^{-2}(t) \tag{6}$$

Where is α_0 the signal gain at unit distance, $d_u^{-2}(t)$ is the Euclidean distance from u to s at the time step t.

computational model

In the system model, the delay of the system includes both the transmission delay and the computational delay. The reason for the calculation delay is that the task is completed by different levels under the multi-level computational model. Contains this formation, the RES layer.

1. Local calculation mode: In the system model, the vehicle unloads some tasks to the RES in each time step. $\tau_{s,u}(t)$ It represents the unloading ratio of the unloading task to the RES and represents the proportion of $\tau_u(t)$ the task performed locally in the vehi $\tau_{s,u}(t)+\tau_u(t)=1$ cle, where. Therefore, the local calculation delay of the vehicle u in the time step is as follows: t

$$D_{\text{local},u}(t) = \tau_u(t) M_u(t) l f_u \tag{7}$$

Where is $M_u(t)$ the data f_u quantity size of the u task, l represents the number of cycles required to process the CPU per unit byte, and represents the computing power of the vehicle u.

2. RES layer calculation mode: Due to the limited calculation resources of vehicle u, when the task is unloaded to the RES layer, the delay includes transmission delay and calculation delay. The transmission delay of the RES layer consists of two parts, one part is that the vehicle u uploads the calculation task to the server s can be expressed as

$$D_{s,u}^{\text{transform}}(t) = \frac{\tau_{s,u}(t) M_u(t)}{R_{s,u}} \tag{8}$$

Where is the task amount accepted by s, and the other part is the delay of the server to transfer the results to the vehicle, which can $M_s(t)$ be expressed as

$$D_{s,u}^{\text{download}}(t) = \frac{M_u^{processed}(t)}{R_{s,u}} \tag{9}$$

Where represents $M_u^{processed}(t)$ the amount of data of the server s responding to the vehicle u. The calculation delay of the RES layer can be expressed as

$$D_{s,u}^{\text{caculate}}(t) = \frac{\tau_{s,u}(t) M_s(t) s}{f_s} \tag{10}$$

Where is the computatio f_s nal power of the server, s. Thus define the server s where the vehicle u unloaded the task and the task delay time that receives the final result can be expressed as.

$$D_{\text{s},u}(t) = D_{s.u}^{\text{transform}}(t) + D_{s.u}^{\text{download}}(t) + D_{s.u}^{\text{caculate}}(t) \tag{11}$$

Cost model

Consider two types of costs in the system: the operation cost includes two parts, respectively, the migration required bandwidth cost and the server-vehicle communication bandwidth cost; the task migration delay cost, that is, the unloading task from the server s1 Migrated to the servers sThe migration latency of the 2s time. To simplify the problem, it is assumed that the start and end times of all tasks are known.

running cost

Operating costs C_{B_m} include migration bandwidth C_{B_s} costs and C_{B_m} communication bandwidth costs. Where the amount of bandwidth is defined as, $\sum_{t\in|T|}\sum_{u\in|U|} B_m^{u,t}$ where:

$$B_{\text{m}}^{u,t} = \varsigma_u \sum_{S_1\in|U|} \sum_{S_2\in(|S|\neq S_1)} j_{S_1,S_2}^{u,t} \tag{12}$$

That is, migration size multiplied by server s ς_u 1 and the server, sThe p $j_{S_1,S_2}^{u,t}$ roportion of migration of 2,

The migration scale represents the time step t task from server s1 Migrated to the servers sThe proportion of the content of the current task, as defined below

$$j_{S_1,S_2}^{u,t} = \sum_{t_1\in\{|T|\le t\}} \sum_{t_2\in(|T|>t)} \frac{h_{S_1,s}^{u,t_1,t_2}}{t_2 - t_1} \tag{13}$$

Similarly, the amount of bandwidth used for the service C_{B_s} is defined as $\sum_{t\in|T|} B_{\text{s}}^{u,t}$, where:Migration delay costs

$$B_s^{\text{u},t} = \sum_{\text{u}\in|U|} \sum_{s\in(|S|)} \tau_{s,u}(t) M_u(t) \tag{14}$$

Take the time experienced when the task start migration starts and the task migration completes as the migration delay. Assuming that each migration task has its maximum delay threshold Y_x^u, and that when the migration cost of the task increases, Y_x^u this part of the

increased cost as the delay violation cost. For example, when the delay exceeds its set threshold. The delay violation cost is, $C_Y^{u,t}$ $C_Y = \sum_{t\in[T]}\sum_{u\in[U]} C_Y^{u,t}$ defined as follows:

$$C_Y^{u,t} = \max(0, Y_r^{u,x} - Y_x^u) D_Y^u \tag{15}$$

$Y_r^{u,x}$ Is the actual delay experienced by the vehicle u at the time step t, which is defined as follows:

$$Y_r^{u,x} = \sum_{S_1\in|S|} \sum_{S_2\in(|S|\neq S_2)} L(S_1, S_2) i_{u,s_1}^t q_{u,s_2}^t \tag{16}$$

L (S1 , S2) of server s1 And sThe delay caused by the communication between the 2. Constant represents the penalty coefficient for migration latency above the threshold. D_Y^u

Total cost of the system

Based on the appeal formula, the total cost formula for the system can be obtained:

$$C_{total} = C_{B_m} + C_{B_s} + C_Y \tag{17}$$

Based on the above model, the optimization problems are summarized as follows in the multi-level computational network system scenario. Under constraints, in order to minimize task latency, we jointly optimize problems such as vehicle movement trajectory prediction, task unloading ratio, and task migration. Reduce latency for all tasks within time T. The final question can be expressed as follows:

$$\min_{\{\tau_{s,u}(t),\ \tau_u(t)\}} = \sum_{t=1}^{T}\sum_{u}^{U}\sum_{s}^{S} \{D_{\text{local},u}(t), D_{s,u}(t)\} \tag{18}$$

$$\text{s.t.} \quad C1: i_{u,s}^t \in [0,1], \forall u, s, t$$

$$C2: \sum_{t=1}^{T}\sum_{t=1}^{T} \tau_{s,u}(t) M_s(t) + \sum_{t=1}^{T}\sum_{t=1}^{T} \tau_u(t) M_u(t) = M$$

$$C3: b_u(t) \in \{0,1\}, \forall t, u$$

$$C4: \tau_{s,u}(t) + \tau_u(t) = 1, \tau_{s,u}(t), \tau_{s,u}(t) \in [0,1]$$

$$C5: C_{total} < C.$$

The constraint C1 indicates that each vehicle can select only select one RES for task calculation and unloading. Constrained C2 ensures that all task calculations are completed over the entire communication period. The constraint C3 represents the congestion of the radio channel between the vehicle u and the server s during a time period. The constraint C4 represents the proportional boundary constrained for task unloading. Constrained condition C5 ensures that the cost of the entire system does not exceed the maximum cost.

2. Solving for the Optimal Path Problem

In this section, the problem (18) is addressed. Show that the problem is NP difficult, modeling this problem as an MDP model,

2.1 Complexity Analysis

First, determine the complexity of the generative migration task in Figure 3.

2.1.1 The number of migration paths

The number of migration paths for a task grows at a rate. $O(S^T)$

Certificate: For each time step t, the vehicle can offload the task to any server. The task must find a task machine to migrate within a certain time step, so the migration path growth rate of the calculation task is. $O(S^T)$

2.1.2 Problem (18) is the NP difficulty

Evidence: because the problem (18) is a broad distribution problem[20] A special case, then it is a NP difficult problem.

Because the migration map grows exponentially for time steps, and the problem (18) is a difficult NP problem. The main difficulty in solving this problem is to generate migration plans for multiple vehicles at the same time.

2.2 MDP Model

To solve the problem (18), the migration plan generation is modeled as an MDP model, and the proposed ANPD-DDPG is used to find a solution.

2.2.1 state space

In the described scenario, RES can be used to obtain real-time information about the environment through smart sensors. At each time step, the system state consists of vehicle location information, obstacle information, task information, task migration information, and migration cost information. The system state of this time period can be expressed as:

$$S_t = i_{u,s}^t\ b_u\ t\ , M_u(t), M_{s,u}(t)\ h_{s_1,s_2}^{u,t_1,t_2}\ C_{\text{total}}\ .$$

(19)

Which represents the position of the vehicle u at the time step time t. Table the congestion condition between the vehicle u and the server s. The presented time step t time u locally calculates the size of the task. This represents the task size of unloading to server s at time step t u. Represents the task server s at the time step time t $i_{u,s}^t$ $b_u(t)$ $M_u(t)$ $M_{s,u}(t)$ h_{s_1,s_2}^{u,t_1,t_2} 1 To sThe migration path of C_{total} 2. It represents the cost that the system has generated. t

2.2.2 action space

When the system performs a lot of operations,

there is a lot of computation, which can lead to significant delays and significant costs. Therefore, to reduce computational complexity, only one server communicates with only one vehicle per decision. The action set in the time step t can be represented as At。It contains the task migration procedure, the server s1 and the server, sThe proportion of task migration completion in 2 and the proportion of vehicle unloading task to RES. Actions in each action space are described as:

$$A_t = \{g_{u,s}^t, j_{s_1,s_2}^{u,t}, \tau_{s,u}(t)\} \quad (20)$$

$g_{u,s}^t$ The task migration process at the current moment, indicating the completion of the migration task at the current moment, $j_{s_1,s_2}^{u,t}$ and indicating the task unloading ratio at the $\tau_{s,u}(t)$ current moment.

2.2.3 reward function

An accurate reward function can be used to aid in the formulation of unloading strategies, while the objective function is usually used as a reward function during machine learning. The objective function is defined in formula (18), so the reward function is set to:

$$R_t = R(S_t, A_t) = -D^{\text{system}}(t) \quad (21)$$

Where is the total system delay $D^{\text{system}}(t)$ of the time step time t, described as follows:

$$D^{\text{system}}(t) = \sum_{k=1}^{K} \max\{D_{\text{local},u}(t), D_{\text{s},u}(t)\} \quad (22)$$

3. ANPD-DDPG Algorithm

In some practical scenarios, you need to ensure the security of the system. Unlike the standard MDP based DDPG algorithm, the MDP based DDPG algorithm only needs to maximize the reward function, thus the system requires dangerous situations in different states when running. To this end, considering the long-term discounted reward (long-term discounted reward) to balance the maximum reward and reduce the cost of risk, the ANPD-DDPG algorithm is designed based on reinforcement learning.

3.1 The constrained Markov Decision Process

Long-term discount (discounted reward) to meet policy π[21] Reward, proposed a CMDP. In the CMDP, $x_n, x_2, \cdots, x_n$ the constraints on the long-term discounted costs are added to the MDP, where the cost function is added to the co x_n nventional MDP. Each is a mapping from the metastatic tuple to the cost. $x_n(\pi) \le u_n$ The goal of CMDP is to select the strategy π to maximize the long-term reward of equation (23) while satisfying the constraints, u_n which is the corresponding limitation. $n \in \{1,2,\cdots,N\}$ Policies can be given by:

$$\pi^* = \arg\max R_t(\pi)$$
$$\text{s.t. } x_n(\pi) \le \text{u}_n, \quad \forall \text{n} \in [1,2,3,\ldots,N]. \quad (23)$$

3.2 ANPD-DDPG Algorithm

In contrast to DQN, the DDPG algorithm mainly solves the prediction problem of continuous action space. The difference in the implementation mainly lies in the choice of the final activation function[22], $k(t) \in [0,\text{K}]$ Whether the action space is continuous or discrete. The actor (actor) network in the DDPG algorithm outputs continuous action, so some action variables need to be processed. While the agent $k(t) = 0$ selects the action variables, a dispersion is required. If, then; if, where indicates the up operation. The task unloa $k^* = 1$ ding scale and speed of the RES are continuous action variables. Is optimized by using all of the above action v $k(t) \ne 0$ ariables together to minimize the system delay. For this purpose, an ANPD-DDPG algorithm was designed to solve the CMDP based on the literature [23]. Unlike DDPG, ANPD-DDPG uses the offline policy data to update the original strategy and the dual variables. $k^* = \lceil k(i) \rceil$ The flow of the ANPD-DDPG algorithm is shown in Algorithm 1.

The whole training process of ANPD-DDPG, is summarized as shown in Figure 4. In contrast to the network structure of the DDPG algorithm, ANPD-DDPG adds a neural network to represent the long-term discount cost. The functional localization neural network of ANPD-DDPG is as follows:

1) Q Online network: according to the current status SSelect t for the current action AT. interacts with the environment to generate the St+1 And, RAnd t and iteratively update the policy network parameter θ.

2) Q Target network: $\theta \to \theta^*$ the next state A sampled from the experience pool (Experience P ool, EP)t+ 1, $\theta_1 \to \theta_1^*$ and, select the next action St+1。

3) Return ω_r the critic online ω_c network:

iteratively update the main $y_t = R_t + \gamma Q_t(S_{t+1} + A_{t+1}, \omega_t)$ network parameters and the target network parameters. And calculate the Q network target value.

4) Target network: $\omega_r \to \omega_r^*$ update and $Q_{t+1}(S_{t+1}, A_{t+1}, \omega_r^*)$ calculate.

5) Cost critic online network: iteratively update Q target network parameters and calculate the value of Q target network. ω_c $Q_t(S_t, A_t, \omega_c)$ $z_t = R_t + \gamma Q_c(S_{t+1} + A_{t+1}, \omega_c)$

6) Cost critic online network: and calculate. $\omega_c \to \omega_c^*$ $Q_{t+1}(S_{t+1}, A_{t+1}, \omega_c^*)$

ANPD-DDPG replication from the online network to the target network is different from the DQN, which directly copies the parameters of the Q online network to the Q target network. ANPD-DDPG selects soft updates, which means that only a fraction of the parameters are updated at a time. Can be expressed as:

$$\theta^* \leftarrow \tau\theta_c + (1-\tau)\theta \tag{24}$$

$$\omega_r^* \leftarrow \tau\omega_r + (1-\tau)\omega_r \tag{25}$$

$$\omega_c^* \leftarrow \tau\omega_c + (1-\tau)\omega_c \tag{26}$$

Where, is the update coefficient usually expressed as small values. At the same time, if the randomness is added in the learning process to increase the coverage of the learning, the final action expression of the interaction with the environment is: τ

$$A_t = \pi_\theta(S_t) \tag{27}$$

To minimize the loss function, the optimizer in the critic network. Similar to DQN, the loss function of the reward critic network can be expressed in the form of mean square error:

$$J(\omega_r) = \frac{1}{\mathrm{m}}\sum_t^m (y_t - Q_r(\Phi(S_t), A_t, \omega_r))^2 \tag{28}$$

Similar cost-critic network loss function:

$$J(\omega_c) = \frac{1}{\mathrm{m}}\sum_t^m (y_t - Q_r(\Phi(S_t), A_t, \omega_c))^2 \tag{29}$$

The ANPD-DDPG performs a deterministic strategy, so that the loss function of the actor network can be expressed as:

$$\nabla_\lambda(\theta, \lambda) = \frac{1}{\mathrm{m}}\sum_t^m \nabla_\pi [Q_r(S_t, A_t, \omega_r)]_{s=s_t, A=\pi\theta(s_t)} - \lambda Q_c(S_t, A_t, \omega_c)_{s=s_t, A=\pi\theta(s_t)} \tag{30}$$

If two different actions A are output for the same state1 And A2, then the two feedback Q values from the critics online network are respectively Q1 And Q2 。 Q1> Q2 means taking the action A1 can be obtained more than in A2 More rewards. Based on the strategy gradient, an A should be addedThe probability of 1 and reduces AThe probability of 2, which means that the network of actors wants to get as much Q as possible. Therefore, the smaller the feedback Q value obtained by the agent, the greater the loss. Therefore, the loss function is calculated as follows:

$$J(\theta) = -\frac{1}{\mathrm{m}}\sum_t^m (Q_r(S_t, A_t, \omega_r) - \lambda Q_c(S_t, A_t, \omega_c) \tag{31}$$

Algorithm 1 ANPD-DDPG algorithm

A) Input: discount factor; learning rate; soft update factor; training period T ω α_1 α_2 τ

B) Initialization: initial target and maximum energy consumption R_0 C_p

Cc) generate the initial task stream; M_u

D) Preprocessing, and obtain the processed task flow; M_u M_u^*

E) initializing an experience pool D of capacity M;

F) Initialize the main network critic parameters and the network parameters of the actors θ ω_r

G) Initialize the critic parameters and the actor home network parameters of the target network θ_1 ω_c

H) Initialize the communication link

i)for do $t = 1, 2, 3\ldots, T$

j) reset and takes the initial state from the CECN function, and S_0 A_0 R_0

k) for do $h = 1, 2, 3\ldots, H$

L) State Normalization: $S_t \to S_t^*$

M) Obtain the action set; $A_t = u(S_t^* | \theta)$

N) execute the action set and obtain the reward function and the next state A_t R_t S_{t+1}

O) if current experience pool D capacity is less than M saves snapshots of current,,, into the experience pool S_t A_t R_t S_{t+1}

p)else

And q) will,,, propose previously existing snapshots in the empirical pool D S_t A_t R_t S_{t+1}

R) Set the target value; $y_t = R_t + \gamma Q_t(S_{t+1} + A_{t+1}, \omega_r^*)$ $z_t = R_t + \gamma Q_c(S_{t+1} + A_{t+1}, \omega_c^*)$

S) Update the parameter sum of the minimum loss function according to formula (25) and (26) ω_r ω_c

T) is updated by formula (24); θ

U) Update the actor family strategy based on formula (27);

And v) update the dual variable of the dual gradient

$$\nabla_{\lambda}(\theta,\ \lambda)=\frac{1}{m}\sum_{t}^{m}\nabla_{\pi}\left[Q_{t}(S_{t},A_{t},\omega_{t})\right]_{S=S_{t}}-u$$

W) Soft update:,,, $\omega_r^*\ \omega_c^*\ \theta^*\ \theta_1^*$

X)end if

Y)end for

Z) Output: Get the actor home network based on formula (31).

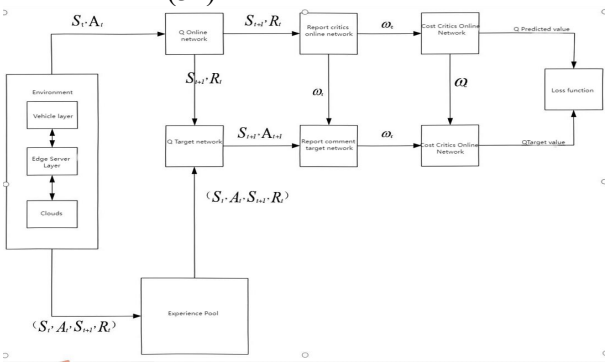


Figure 4. ANPD-DDPG Algorithm Flow

4. Simulation Experiment

In this part, a series of simulation experiments are performed to verify the algorithm performance of the proposed ANPD-DDPG algorithm under the proposed multi-level network architecture.

4.1 Environment Setting

In the simulation experimental system, a multi-level computing architecture of the square area is set, where there is a uniform random distribution area and a cloud server, which is uniform and randomly distributed edge service. All the vehicles in the experiment were only communicating with the RES, and all the RESs were connected to the cloud server, as shown in Figure 1. The initial position of the vehicle was randomly drawn from a uniform distribution.

Vehicle movement was predicted using a Markov model. Table 1 shows the accuracy of the Markov model used for simulation for vehicle position prediction. As the data show, locations with larger time steps reduced the prediction accuracy. The more servers can predict the vehicle location more difficult.

Table 1. Mobile Model Predicts the Average Probability of the Server Closest to Vehicle U

time step	1	5	10	20
5 Servers	0.87	0.62	0.49	0.26
10 Servers	0.80	0.49	0.39	0.14
20 Servicers	0.71	0.40	0.25	0.13

The complexity of the migration path. As shown in Table 2, the number of edges in the migration graph increases geometrically with time and servers in the migration graph.

Table 2 grows with the server and the time. Change of the edge number of the migration map

Table 2. With the Increase of Servers and Time. Change in the Number of Edges in the Migration Graph

		Server number			
		5	10	15	20
time step	5	172	576	1042	2438
	10	538	6190	11356	20378
	25	2041	9166	16736	27624

To assess the impact of resource constraints, the simulations were validated using limited and abundant resources. The RES resources for many edge computing systems are limited because edge devices often have limited hardware (i. e., smart cameras and durable laptops). In the finite resource setting, the resource capacity is drawn from a uniform distribution, based on the total number of vehicles in the system, each link can migrate six computational tasks within a single time step. Resource constraints do not affect migration decisions for abundant resources.

4.2 Assessment of the Algorithm Convergence Properties

Figure 5 describes the convergence of the proposed ANPD-DDPG algorithm. Figure 5 shows that the system delay starts to decline from 0 to 200 iterations, and the convergence stability reaches at around 200 iterations. The initial delay fluctuation is due to the large gap between the initial value of the Lagrange factor and the optimal, the action selection in the algorithm makes the delay decrease rapidly, and with the increase of iterations, the system gradually approaches the optimal delay performance.

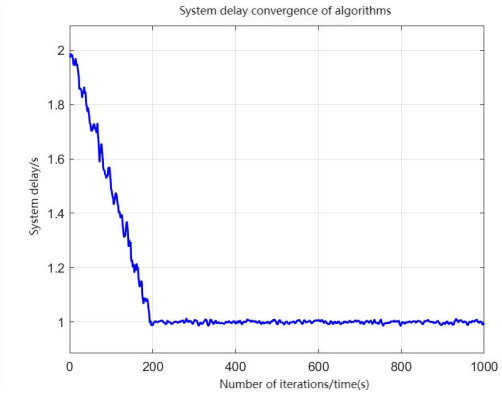


Figure 5. Algorithm Convergence

4.3 Comparison of the different migration methods

The ANPD-DDPG algorithm and the three different migration methods were performed. Naive method (naive)[24] Select the closest available server without migration to minimize the cost, which is similar to SDN / NFV placement[25] Optimization. Shortness-view (myopic)The [26] method migrate tasks to the nearest feasible server at each time step, which is similar to the reactive migration framework[27]; Cloud method[28] Place all the tasks that need to migrate to the cloud server, regardless of resource constraints, similar to central processing. The value of the vehicle trajectory prediction was verified by this comparison.

As shown in Figure 6, comparing the task migration costs implemented by different migration scenarios with limited and sufficient resources, all plan generation methods can reduce costs with abundant resources compared with limited resources, as each unloading task can achieve low-latency placement. The ANPD-DDPG algorithm method significantly outperforms the other three algorithms due to the small number of task migrations. Especially with limited resources, the ANPD-DDPG algorithm saves 33% of the cost relative to the naive method and 18% from the short-sighted method. With limited resources, the ANPD-DDPG algorithm is better than the cloud mode, while with sufficient resources, because the unloading task can be fully offloaded to each RES, their performance is equivalent

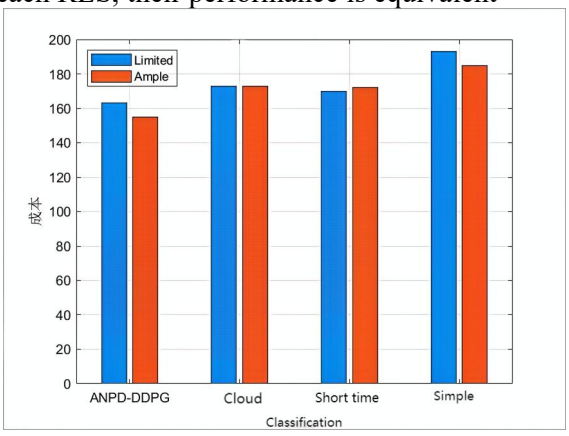


Figure 6. Migration Costs Under Different Resources

Figure 7 shows the systematic delay cost of the ANPD-DDPG algorithm and the three migration schemes. The ANPD-DDPG algorithm outperforms the other three schemes due to more frequent task offloading and lower system latency. The naive approach does not migrate, and task migration begins to be affected as the vehicle moves away from its original location. The latency costs generated by myopic methods are higher than those generated by ANPD-DDPG and cloud methods (48% and 39%, respectively) because of frequent migrations and higher placement costs (18% and 16%, respectively), and bandwidth usage costs (approximately 380% for SG and cloud methods). Because the ANPD-DDPG algorithm provides better vehicle movement prediction by updating its conditions for a Markov chain model of vehicle movement, the ANPD-DDPG algorithm improves performance by 10% and 15% relative to the naive and cloud methods, respectively.

Cost System latency Deploy cost Bandwidth Cost classification cost

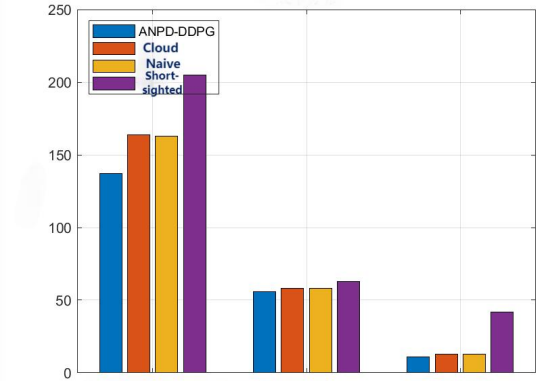


Figure 7. Migration Costs of Various Solutions

System delay: Figure 8 shows the influence of the ANPD-DDPG algorithm and the other three schemes on the system delay under different vehicle densities, In Figure Figure 88, It can be seen that when the density of the square system vehicles increases, The system delay of the cloud solution remains the same, While the system delay of ANPD-DDPG algorithm, naive and short-sighted schemes, The time delay increases, This is due to the increased vehicle density reducing the transmission efficiency between the vehicle and the RES, In turn leads to increased system delay, And when the vehicle density reaches 0.3, Both naive and cloud solutions have almost the same delay, This is due to the increasing number of unloading task migration in more naive vehicle scenarios, Thus affecting the system time delay. However, due to the ANPD-DDPG algorithm, the system delay performance is obviously due

to the other three schemes. In addition, the density of vehicles has significantly more effects on the naive and cloud methods than the ANPD-DDPG algorithm, thus verifying the robustness of ANPD-DDPG in the case of high vehicle density.

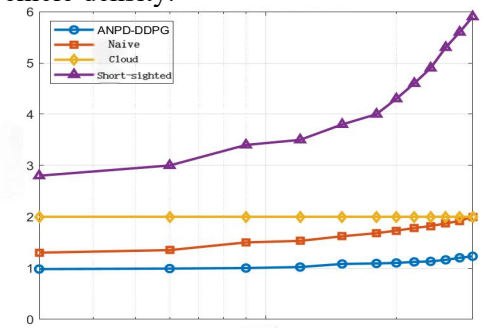
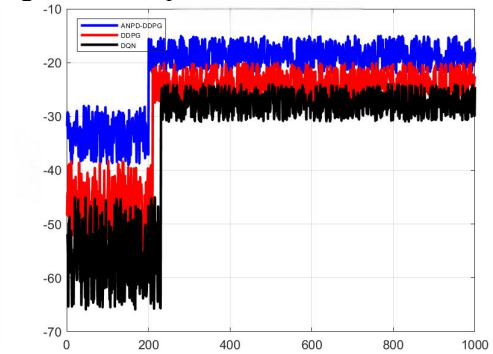


Figure 8. System Latency for Different Solutions

4.4 Efficiency of the Algorithm

Figure 9 shows the performance results between the ANPD-DDPG algorithm and the other baseline algorithms. The total number of iterations is 1000, and DQN, DDPG and ANPD-DDPG all converge when increasing the number of iterations. The results show that the ANPD-DDPG utility significantly outperforms the other two algorithms, since all three algorithms contain one policy network and one target network. The dual network structure can find the best action strategy by isolating the correlations between the training data. Unlike the DQN algorithm, the ANPD-DDPG algorithm can output continuous actions, which is more advantageous in choosing the action space. The results show that the converged results of the ANPD-DDPG algorithm are significantly better than the delayed results of both.

Convergence ofrewards between different algorithms Report



Number of iterations

Figure 9. Convergence Rate for Different Algorithms

Figure 10 shows, the average delay size of the ANPD-DDPG algorithm, the DQN algorithm, and the DDPG for different task sizes. In the figure, the total latency of the ANPD-DDPG algorithm is significantly lower than the DQN and DDPG algorithm algorithms for the same size task, because the non-negligible space between the discrete action space and the available actions needs to be explored. It is difficult to find a better unloading strategy in the DQN algorithm. Furthermore, ANPD-DDPG can explore the continuous action space to obtain suitable strategies by taking precise actions. The ANPD-DDPG algorithm grows much slower than the other schemes, which further demonstrates the advantages of the ANPD-DDPG algorithm. Figure Figure 11 shows the total delay at different transmission power levels. Demonstrated, the advantages of the ANPD-DDPG algorithm.System latency under different task sizesSystemtotal delay

Total task size

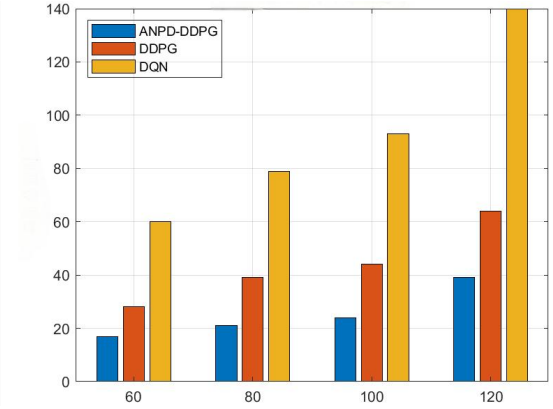
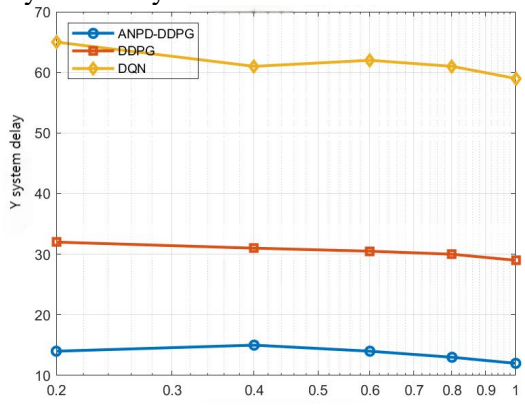


Figure 10. System Latency Under Different Task Sizes

System delay at different transmission rates



Uplink transmission rate of X

Figure 11. System Delay at Different Transmission Rates

5. Conclusion

This paper proposes a multi-level computational framework applied to task unloading and migration, which performs vehicle trajectory prediction through MMO model, and solves the bandwidth and delay pressure caused by a large amount of historical data derived from vehicles in traditional machine learning for trajectory prediction. The ANPD-DDPG algorithm is proposed to develop a reasonable migration plan. By analyzing the ANPD-DDPG algorithm, compared with the other three migration schemes, ANPD-DDPG has a good performance in reducing the cost of task migration and improving the system response time. However, the text considers a scenario in which a RES servers for a vehicle. In the future, we can study how to generate migration task plans through servers for multiple vehicles on multiple edge nodes simultaneously.

References

[1] DAI Y, Xyu D, MAHARJAN S, et al. Joint load balancing and offloading in vehicular edge computing and networks[J]. IEEE Internet of Things Journal, 2018, 6(3): 4377-4387.

[2] Tamani N, Brik B, Lagraa N, et al. On link stability metric and fuzzy quantification for service selection in mobile vehicular cloud[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 21(5): 2050-2062.

[3] Yang C, Liu Y, Chen X, et al. Efficient mobility-aware task offloading for vehicular edge computing networks[J]. IEEE Access, 2019, 7: 26652-26664.

[4] Sorkhoh I, Ebrahimi D, Atallah R, et al. Workload scheduling in vehicular networks with edge cloud capabilities[J]. IEEE Transactions on Vehicular Technology, 2019, 68(9): 8472-8486.

[5] Wang L, Jiao L, Li J, et al. MOERA: Mobility-agnostic online resource allocation for edge computing[J]. IEEE Transactions on Mobile Computing, 2018, 18(8): 1843-1856.

[6] Zhang Yilin, Liang Yuzhu, Yin Mujun, et al. Survey on the Methods of Computation Offloading in Mobile Edge Computing[J]. Journal of Computer Science, 2021, 44(12): 2406-2430.)

[7] Liu J, Mao Y, Zhang J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//2016 IEEE international symposium on information theory (ISIT). IEEE, 2016: 1451-1455.

[8] You C., Huang K. "Multiuser resource allocation for mobile edge computation offloading." Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, USA, 2016, pp. 1-6.

[9] Zhao P, Tian H, Qin C, et al. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing[J]. IEEE access, 2017, 5: 11255-11268.

[10] Wang S, Urgaonkar R, Zafer M, et al. Dynamic service migration in mobile edge computing based on Markov decision process[J]. IEEE/ACM Transactions on Networking, 2019, 27(3): 1272-1288.

[11] Wang J, Hu J, Min G, et al. Online service migration in edge computing with incomplete information: A deep recurrent actor-critic method[J]. arXiv preprint arXiv:2012.08679, 2020: 41.

[12] Wang S, Guo Y, Zhang N, et al. Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2019, 20(3): 939-951.

[13] Labriji I, Meneghello F, Cecchinato D, et al. Mobility aware and dynamic migration of MEC services for the Internet of Vehicles[J]. IEEE Transactions on Network and Service Management, 2021, 18(1): 570-584.

[14] Ouyang T, Zhou Z, Chen X. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2333-2345.

[15] Gebrie H, Farooq H, Imran A. What machine learning predictor performs best for mobility prediction in cellular networks?[C]//2019 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2019: 1-6.

[16] Al-Shaery A M, Ahmed S G, Aljassmi H, et al. Open Dataset for Predicting Pilgrim Activities for Crowd Management During

Hajj Using Wearable Sensors[J]. IEEE Access, 2024.
[17] Kim T, Sathyanarayana S D, Chen S, et al. Modems: Optimizing edge computing migrations for user mobility[J]. IEEE Journal on Selected Areas in Communications, 2022, 41(3): 675-689.
[18] Kim T, Sathyanarayana S D, Chen S, et al. Modems: Optimizing edge computing migrations for user mobility[J]. IEEE Journal on Selected Areas in Communications, 2022, 41(3): 675-689.
[19] IMANE A. Mobile Edge Computing for the Internet of Things[J]. 2019.
[20] Rejiba Z, Masip-Bruin X, Marín-Tordera E. A survey on mobility-induced service migration in the fog, edge, and related computing paradigms[J]. ACM Computing Surveys (CSUR), 2019, 52(5): 1-33.
[21] Liang Z, Liu Y, Lok T M, et al. Multi-cell mobile edge computing: Joint service migration and resource allocation[J]. IEEE Transactions on Wireless Communications, 2021, 20(9): 5898-5912.
[22] Ngo M V, Luo T, Hoang H T, et al. Coordinated container migration and base station handover in mobile edge computing[C]//GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, 2020: 1-6.
[23] Ma L, Yi S, Carter N, et al. Efficient live migration of edge services leveraging container layered storage[J]. IEEE Transactions on Mobile Computing, 2018, 18(9): 2020-2033.
[24] Xu Xiaobin, Wang Qi, Fan Cunqun, et al. An Aggregated Edge Computing Resource Management Method forSpace-Air-Ground Integrated Information Networks[J]. Journal of Computer Science, 2023, 46(04): 690-710.)
[25] Kuang Zhufang, Chen Qinglin, Li Linfeng, et al. Multi-user Edge Computing Task offloading Scheduling and ResourceAllocation Based on Deep Reinforcement Learning[J]. Journal of Computer Science, 2022, 45(04): 812-824.)
[26] Wang S, Urgaonkar R, Zafer M, et al. Dynamic service migration in mobile edge computing based on Markov decision process[J]. IEEE/ACM Transactions on Networking, 2019, 27(3): 1272-1288.
[27] Zeng Yaoping, Jiang Weiwei, Liu Yueqiang, et al. Dynamic Offloading Algorithm for Tasks in Vehicle Edge Networks [J]. Computer Engineering and Applications, 2024, 60(14): 267-274.)
[28] Labriji I, Meneghello F, Cecchinato D, et al. Mobility aware and dynamic migration of MEC services for the Internet of Vehicles[J]. IEEE Transactions on Network and Service Management, 2021, 18(1): 570-584.
[29] Ouyang T, Zhou Z, Chen X. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(10): 2333-2345.
[30] Cao T, Qian Z, Wu K, et al. Service placement and bandwidth allocation for MEC-enabled mobile cloud gaming[C]//2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM). IEEE, 2021: 179-188.
[31] Yang B, Chai W K, Xu Z, et al. Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications[J]. IEEE Transactions on Network nd Service Management, 2018, 15(1): 475-488.
[32] Fallah S N, Ganjkhani M, Shamshirband S, et al. Computational intelligence on short-term load forecasting: A methodological overview[J]. Energies, 2019, 12(3): 393.
[33] Hu B, Hu W. Linkshare: Device-centric control for concurrent and continuous mobile-cloud interactions[C]//Proceedings of the 4th ACM/IEEE Symposium on Edge Computing. 2019: 15-29.
[34] Xu Xiaolong, Fang Zijie, Qi Lianyong, et al. A Deep Reinforcement Learning-Based Distributed Service Offloading Method for Edge Computing Empowered Internet of Vehicles[J]. Journal of Computer Science, 2021, 44(12): 2382-2405.)
[35] Ottenwälder B, Koldehofe B, Rothermel K, et al. MCEP: A mobility-aware complex event processing system[J]. ACM Transactions on internet technology (TOIT), 2014, 14(1): 1-24.
[36] Jia Y, Wu C, Li Z, et al. Online scaling of NFV service chains across geo-distributed datacenters[J]. IEEE/ACM Transactions on

Networking, 2018, 26(2): 699-710.
[37] Bari F, Chowdhury S R, Ahmed R, et al. Orchestrating virtualized network functions[J]. IEEE Transactions on Network and Service Management, 2016, 13(4): 725-739.
[38] Pande S K, Panda S K, Das S. Dynamic service migration and resource management for vehicular clouds[J]. Journal of Ambient Intelligence and Humanized Computing, 2021, 12: 1227-1247

