Journal of Computers Vol. 35 No. 1, February 2024, pp. 243-254
doi: 10.53106/199115992024023501020

Research on Vehicle Task Management Based on TBMADDPG

Zhao-Nian Li[1], Yao-Chen Zhang[2], Zhen-Jiang Zhang[1*], Wen-Hui Wang[1], Mao-Jie Zhang[3], Guo-Hua Shi[3]

[1] Electronics and Information Engineering, Beijing Jiaotong University, China
[2] The enterprise development department of Inspur Software Technology Co., Ltd, China
[3] Xincheng Information Technology Co., Ltd, China

965317502@qq.com, zyc@inspur.com, zhangzhenjiang@bjtu.edu.cn,
23111039@bjtu.edu.cn, zhangmj@hbxcxx.com, shigh@hbxcxx.com

Received 15 December 2023; Revised 15 January 2024; Accepted 31 January 2024

Abstract. Recently, the traditional cloud computing network of vehicle networking has some problems to be solved: 1) the security trust between the vehicle and the subgrade unit; 2) The vehicle may be attacked by potentially malicious edge servers during task unloading. In this paper, in order to solve the above problems, aiming at the security problem of the edge computing network in the vehicle task unloading and resource allocation problems of multi-vehicle and multi-roadbed units in the urban intersection scene, the vehicle task unloading and resource management optimization algorithm and trust model based on the vehicle edge computing network are constructed, and the approximate optimal simulation is carried out for the urban intersection scene. Simulation results show that the proposed algorithm can effectively improve the overall efficiency of the system.

Keywords: internet of vehicles, edge computing, resource allocation, trust model, reinforcement learning

1 Introduction

Nowadays, the introduction of edge computing unit in vehicle networking has some problems to be solved in practical application scenarios. The first is the security and trust issues between the vehicle and the subgrade unit. When the malicious subgrade unit maliciously interferes with the vehicle, such as the malicious disclosure of the vehicle user's private data, location and other information, it will greatly affect the service quality of the vehicle edge computing network. On the other hand, the computing resources and energy resources in the roadbed processing unit are relatively limited and dynamic, so when the vehicle is attacked by malicious edge servers during the unloading process, the resource competition between vehicles will be more intense and unstable.

Based on the analysis of the trust problem and resource allocation problem between vehicles and roadbed units in road intersection scenarios, this paper proposes a distributed resource scheduling algorithm TBMADDPG based on trust model and deep reinforcement learning. Under the premise of ensuring the constraints of vehicle task unloading and resource allocation scheme, Minimize the time and energy cost consumed by the vehicle to process the task, and ensure that the vehicle in the system can complete the task calculation or unloading in a trusted environment. Finally, the reliability and effectiveness of the proposed algorithm are verified by the simu-lation comparison with other existing schemes.

2 Related Works

In recent years, the global trust model analysis based on direct trust and indirect trust in the scenario of vehicle networking is diverse. The distributed trust mechanism model proposed by Mohri et al. comprehensively considers the information interaction between vehicles and roadbed units and the decay of trust over time and other factors [1]. The study [2] introduced timestamp mechanisms and blockchain concepts into trust management systems. Manickavasagam et al. uses Mutual Trust Model (MTM) for Wireless Body Area Network (WBAN) with the help of Fog-Node (FN) to address these issues and to ensure the trustworthiness of the information acquired [3].

* Corresponding Author

In the research of edge computing, data security, trusted evaluation and other security issues are also widely discussed. In recent years, many peer-to-peer trust models have emerged. For example, the intrinsic trust model [4] is one of the first models to dynamically calculate the global trust of nodes. The model definition calculates and provides feedback on the local trust values of nodes after completing information exchange, in order to calculate global trust. However, this algorithm suffers from high complexity and difficulty in convergence when applied to large-scale networks. For the vehicle edge computing scenario, Hu et al. proposed a distributed reputation management mechanism to ensure the information security in VEC, and updated the global reputation information through the weighted subjective logic [5]. In order to simultaneously respond to malicious server attacks and improve computational efficiency, Zhu et al. used a risk model and information entropy to describe the direct and indirect trust between nodes in the system, and constructed a lightweight trust mechanism with multi-source feedback [6]. In order to reduce the risk of topology attacks on the central controller of SDN in the scenario of centralized management of information within VEC, Ouyang et al. proposed an attack tolerance scheme for VEC networks that can self recover to a certain extent using deep reinforcement learning under existing defense mechanisms [7].

Regardless of the work scenario, task offloading and resource scheduling under trusted conditions have strong practical significance, so research on trust problems is also very hot. Compared with traditional deep reinforcement learning algorithms, the effective privacy aware task offloading strategy proposed by Rago et al. based on deep decision states and privacy vulnerabilities in new locations has a more reliable learning speed and effectiveness [8]. The direct trust degree is used to judge edge nodes, and the reverse auction model is used to provide edge side cache services. Zheng et al. have effectively combined the trust problem in edge computing with the computing cache problem, improving the overall service performance of the edge network [9]. When addressing privacy protection and computational cost optimization strategies, Wang et al. introduced Lyapunov optimization functions to ensure that users maintain privacy and security while optimizing their user experience in the face of inference attacks [10].

The analysis of global trust models based on direct trust and indirect trust in the context of connected vehicles is also diverse. Zhang et al. propose a trust management system of IoV based on blockchain, which formalizes a complete vehicle reputation value calculation scheme to deal with the problem of calculating the credibility of messages [11]. In order to maintain stable interconnection of vehicle nodes and prevent the spread of false information, research [12] introduces timestamp mechanisms and blockchain concepts into trust management systems. Process the sorted vehicle information based on role-based strategies, priority, and thread concepts, and calculate the direct trust value; Indirect trust is estimated and calculated using the method of similarity measurement. Liu et al. used the satisfaction function of services and the degradation factor function of time to improve the direct trust value described by the Bayesian equation, and then calculated the weights of indirect trust between various devices. They improved the accuracy of the indirect trust model through an improved grayscale correlation method [13]. In addition, Du et al. calculated the server load, service rejection rate, and service access delay as trust indicators [14]; Tan et al. proposed that nodes only need to calculate direct trust values, while indirect trust values are obtained from Sink, thereby reducing energy consumption in iterative calculations [15]; Hui et al. converted the vehicle node benefits brought by punishing attack behavior and rewarding normal behavior nodes in the game into node trust values for trust evaluation [16]; Su et al. evaluated and inferred the collected trust evidence to provide more accurate trust evaluation results for the selection of unloading strategies in the future [17].

With the popularization of machine learning and its potential in the field of Internet of Things applications, more and more researchers are applying this technology to optimize task offloading and resource scheduling strategies. For example, in the study [18-19], the LSTM algorithm was used to dynamically predict the edge communication and computing resources of mobile users, and decisions were made based on the predicted future data. In terms of simulation optimization decision, Xu et al. used the deep Q network to create an intelligent resource allocation scheme for edge computing scenarios with variable resource environments to achieve the effect of optimal delay [20]. Methods based on deep Q-networks, such as DQN, double DQN, etc., have high costs in solving multi discrete offloading action problems. Therefore, policy based methods, such as AC-DRL and deep deterministic policy gradient algorithms, have been successively applied in current research [21]. In addition, when facing the challenge of vehicle computing intensive applications, utilizing multi-agent training to reduce instability in the environment and ensure policy updates [22].

3 System Model

3.1 Task Model

The execution of intelligent tasks in the industrial internet needs to be kept complete, but due to the fact that intelligent devices may reach multiple tasks at a certain time slot t, intelligent devices as CRC can offload some tasks to ESC or intelligent devices as CRP for processing. The task type is represented by $K = \{1, 2, \ldots, K\}$, where $K \leq M$, because there will be multiple smart devices processing the same type of task, the number of smart devices is always greater than or equal to the number of task types. The type of task processed by the intelligent device is represented by $k(M) = \{k_1, k_2, \ldots, k_M\}$, and for any task processed by the intelligent device m, $k_m \in K$.

For task $\forall k \in \mathcal{K}$, there are two parameters: the computational resources required by the task, i.e. the number of CPU cycles, and the size of the input data for the task (in bits). Two vectors, $(M) = \{d_1, d_2, \ldots, d_M\}$ and $b(M) = \{b_1, b_2, \ldots, b_M\}$, can be used to represent the required computing resources and input data size for all devices to execute tasks.

At the same time, the task arrival rate of different smart devices is different and time-varying, using A_t^m represents the number of task arrivals of intelligent device m in time slot t, vector $A_t = \{A^1_t, A^2_t, \ldots, A^M_t\}^T$ represents the number of device tasks that have arrived at time slot t.

Intelligent devices need to unload tasks to other devices for better performance due to the limitations of their own computing resources when processing tasks. The model modeling and unloading decision constraints for edge computing task unloading and D2D task unloading are as follows.

Edge Computing Task Unloading. Whether device m unload tasks to ESC during time slot t is indicated by $\alpha_t^m \in \{0,1\}$, $\alpha_t^m = 0$ indicates that device m does not unload tasks to ESC during time slot t, $\alpha_t^m = 1$ indicates that device m offloads tasks to ESC during time slot t. Is all devices performing task offloading to ESC represented as $\alpha_t = \{a_t^1, a_t^2, \ldots, a_t^M\}^T$.

D2D Unloading. Whether device m unload tasks to device n during time slot t is indicated by $\beta_t^{m,n} \in \{0,1\}$, $\beta_t^{m,n} = 0$ means that device m does not unload tasks to device n, $\beta_t^{m,n} = 1$ indicates that device m is unloading tasks from device n. Special, $\beta_t^{m,m} = 1$ indicates that the role of device m is CRP, and local tasks are completely executed locally; and $\beta_t^{m,m} = 0$ indicates that the device m role will be partially or completely uninstalled as a CRC local task. From the above definition, it can be seen that the unloading decision of time slot t intelligent device m regarding D2D can be expressed as

$$\beta_t^m = \{\beta_t^{m,1}, \beta_t^{m,2}, \ldots, \beta_t^{m,M}\}^T, m \in M. \tag{1}$$

So the decision to perform D2D offloading between all devices can be expressed as:

$$\beta_t = \{\beta_t^1, \beta_t^2, \ldots, \beta_t^M\} = \begin{bmatrix} \beta_t^{1,1} & \beta_t^{2,1} & \cdots & \beta_t^{M,1} \\ \beta_t^{1,2} & \beta_t^{2,1} & \cdots & \beta_t^{M,2} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_t^{1,M} & \beta_t^{2,1} & \cdots & \beta_t^{M,M} \end{bmatrix}. \tag{2}$$

Unloading Decision Constraints. For $\forall m \in M$, there are three scenarios for device m's offloading decision: 1. As a CRP, the local task is completely executed locally and may provide computing resources for other devices; 2. As CRC, unload part or all of the local tasks to ESC for execution, that is, edge computing unloading; 3. As a CRC, partially or completely offloading local tasks to CRP for execution, i.e. D2D offloading. So for $\forall t \in T$, $\forall m \in M$, the unloading of device m time slot t needs to meet the following constraints:

$$\alpha_t^m + \sum_{n=1}^{M} \beta_t^{m,n} = 1, m \in M. \tag{3}$$

At the same time, it should be considered that when performing D2D offloading for these computing tasks, it can be seen from the above problem description that it is necessary to consider the heterogeneity of different smart devices and the limited storage capacity of the devices. That is, only smart devices that perform the same computing task can perform D2D offloading between them. The task relationship between smart device m and other smart devices can be expressed as:

$$R^m = \{R^{m,1}, R^{m,2}, ..., R^{m,M}\}, m \in M. \tag{4}$$

For $\forall\ m, n \in M$, $R^{m,\,n} \in \{0,1\}$, $R^{m,\,n} = 1$ indicates that the two devices have the same task type and can be unloaded with D2D, while when R ^ (m, n) = 0, it indicates that the two devices have different task types and cannot be unloaded with D2D. The task relationship matrix between all smart devices can be represented as:

$$R = \{R^1, R^2, ..., R^M\} = \begin{bmatrix} R^{1,1} & R^{2,1} & \cdots & R^{M,1} \\ R^{1,2} & R^{2,2} & \cdots & R^{M,2} \\ \vdots & \vdots & \cdots & \vdots \\ R^{1,M} & R^{2,M} & \cdots & R^{M,M} \end{bmatrix}. \tag{5}$$

Considering the task relationships between different devices, it is necessary to uninstall between intelligent devices of the same task type during D2D uninstallation, which satisfies the following constraints:

$$\sum_{n=1,n\neq m}^{M} R^{m,n} * \beta_t^{m,n} = 1 \quad if\ \ \alpha_t^m = 0\ and\ \beta_t^{m,m} \neq 1. \tag{6}$$

Simultaneously considering $\beta_t^{m,n} = 1$, according to the constraint of equation 4-3, for $\neq m$, $\forall\ n \in M$, there are $\beta_t^{m,n} = 0$ and equations 4-6 still hold, then the constraint can be simplified as:

$$\sum_{n=1}^{M} R^{m,n} * \beta_t^{m,n} = 1,\ m \in M \quad if\ \ \alpha_t^m = \#. \tag{7}$$

Considering the weak communication capability of smart devices themselves, only one CRP can serve at most one CRC in the same time slot. The constraint form is as follows:

$$\sum_{m=1,m\neq n}^{M} \beta_t^{m,n} \leq 1,\ n \in M. \tag{8}$$

3.2 Communication Model

Affected by a series of factors including time delay, energy consumption, communication channel status, vehicle speed and so on, the tasks generated by vehicles will be effectively unloaded and calculated with different strategies in the vehicle edge computing network. Affected by the unloading decision $X = \{0, 1, 2\}$, the task unloading behavior of vehicles in each roadbed unit area can be represented as $A = \{a_1, a_2, \ldots, a_n\}$. In the offloading decision set, decision 0 indicates that the vehicle task is completed locally, decision 1 indicates that the task is offloaded to the RSU to which the vehicle belongs, and decision 2 indicates that the task sought assistance from neighboring RSUs during the offloading calculation process.

In this chapter, due to the application of Massive MIMO technology, the roadbed unit can simultaneously interact with multiple vehicles for information exchange [23]. In order to simplify the model and facilitate calculations, the communication channel bandwidth between different vehicles and different roadbed units is set to be equal and represented by parameters. Meanwhile, use $x = 1$ to indicate that the task has been uninstalled; Use $x = 0$ to indicate that the task has not been uninstalled.

In practical scenarios, the state of communication channels can also be affected by noise. According to Shannon's formula, the transmission rate of information in wireless channels can be calculated as:

$$R_n = w \cdot \log_2\left(1+\frac{p_n h_n}{\sigma^2 + I_n^{RSU}}\right), forn \in N. \tag{9}$$

3.3 Computation Model

For tasks of task type φ_1, only the vehicle's local execution computational model needs to be considered. Each car is equipped with a CPU processor to process or calculate tasks generated by the vehicle. In order to handle the tasks Y_n which are generated by calculations, each vehicle needs to allocate computing resources that comply with the allocation restriction policy $F = \left\{ f_n^{loc} \middle| 0 < f_n^{loc} \le F_n^{loc}, n \in N \right\}$. In addition, due to the fact that tasks of the type φ_1 are only calculated locally on the vehicle and do not involve behaviors such as information transmission, the execution time and energy consumption of tasks locally can be represented as follows:

$$t_1 = \frac{C_n}{f^{loc}}. \tag{10}$$

$$E_1 = K\left(f_n^{loc}\right)^2 C_n. \tag{11}$$

Among them, the parameter K is the power parameter, and its size only depends on the architecture of the CPU itself.

For tasks of task type φ_2, it is necessary to consider two models: computing the task locally on the vehicle and offloading it to the corresponding RSU for computation. The task model calculated locally on the vehicle can continue to use the model discussed earlier; In the uninstallation model, it is necessary to consider the process of uploading and downloading tasks, but due to the size of the task information itself, the calculation return result is generally considered very small. Therefore, the time and energy consumption caused by feedback information can be ignored. So, the time consumption for unloading and calculating tasks of type φ_2 includes the calculation time in the roadbed unit and the upload task time. Therefore, the time consumption that this type of task may bring is defined as:

$$t_2 = \begin{cases} t_2^{loc} = \dfrac{C_n}{f_n^{loc}} \\ t_2^{RSU} = t_2^{exu} + t_2^{up} = \dfrac{C_n}{f^{RSU}} + \dfrac{D_n}{R_n} \end{cases}. \tag{12}$$

The energy consumption of unloading tasks only needs to consider the energy consumed by vehicles during the upload process, so the energy consumed by vehicles to complete tasks of type φ_2 can be defined as:

$$E_2 = \begin{cases} E_2^{loc} = K(f_n^{loc})^2 C_n \\ E_2^{RSU} = p_n^k \dfrac{D_n}{R_n} \end{cases}. \tag{13}$$

For tasks of task type φ_3 , it is necessary to consider two models: computing the task locally on the vehicle and offloading it to the corresponding RSU for computation. However, due to the change in task type, the calculation result data of this type of task cannot be ignored, and its size is η times the size of the task. In addition, as the information transmission between two adjacent roadbed processing units is carried out through optical fibers, this chapter ignores the transmission delay of this process. According to the above model, the formulas for defining the time and energy consumption of tasks with task type φ_3 are:

$$t_3 = \begin{cases} t_3^{RSU} = t_3^{exu} + t_3^{up} + t_3^{down} = \dfrac{C_n}{f^{RSU}} + (1+\eta)\dfrac{D_n}{R_n} \\ t_3^{adj_{RSU}} = \dfrac{C_n}{f^{adj_{RSU}}} + (1+\eta)\dfrac{D_n}{R_n} \end{cases}. \quad \textbf{(14)}$$

$$E_3 = E_3^{RSU} = E_3^{adj_{RSU}} = (1+\eta) p_n^k \frac{D_n}{R_n}. \quad \textbf{(15)}$$

When calculating tasks generated by the vehicle itself, the task offloading strategy is $a_t^{i,0} = 1$, and the time and energy consumption generated during this period can be calculated using formulas (10) and (11).

3.4 Modeling Design

In the vehicle network, it is assumed that the intersection has two types of subgrade service units, including normal and malicious, and are connected to each other through optical fibers. Normal subgrade units can provide computing resources and secure processing environment, malicious subgrade units will provide false service information and disclose the private data of the vehicle. In order to improve the model's ability to identify malicious subgrade units and balance the allocation of edge resources, this paper proposes a distributed resource scheduling algorithm TBMADDPG based on trusted model and deep reinforcement learning, so that the system can effectively and quickly identify malicious subgrade units while reducing the overall time and energy consumption.

At the initial time, each vehicle initializes the trust of all subgrade units in the model, and then updates the trust of each time slot according to the analysis in the previous section. For all vehicles and roadbed units, the global trust matrix can be expressed as:

$$TrustMatrix(t) = \begin{bmatrix} Trust_{1,1}(t) & \cdots & Trust_{1,m}(t) \\ \vdots & \ddots & \vdots \\ Trust_{n,1}(t) & \cdots & Trust_{n,m}(t) \end{bmatrix}. \quad \textbf{(16)}$$

In the description of the trust degree of the subgrade unit, the trust coefficient of the subgrade unit in a certain time slot is defined as the average of the global trust value of all vehicles, which can be expressed as:

$$Trust_j(t) = \frac{\sum_{i=1}^{n} Trust_{i,j}(t)}{n}. \quad \textbf{(17)}$$

In the model, the vehicles will be randomly generated at the beginning of each time slot time of different types of tasks, at the same time subgrade unit to provide vehicles also dynamic change of computing resources, the allocation rate notes for T_R^t. In addition, the global trust matrix of the model is updated according to the interaction between the vehicle and the subgrade unit in each time slot. According to whether the information belongs to the vehicle's own information or the global information, the vehicle's own state information is first defined as:

$$s_i(t) = \left[\tau r, i_{w,i_{\max}}\right]. \quad \textbf{(18)}$$

The system state space of the current timeslot *t* can be defined as:

$$S_i = \left(s_1(t), s_2(t), ..., s_n(t), t_{w,m}, T_R^t, TrustMatrix\right). \quad \textbf{(19)}$$

After obtaining the environment state space, the next action is selected according to the policy in the policy network, and the next state space is obtained. Since all the vehicle's task is not completed by the vehicle itself is unloaded to the calculation model of a calculated in subgrade unit, so the vehicle *i* in current time slot *t* uninstall decisions can be expressed as a (m + 1) × 1 matrix. So, the system dynamic space can be finally expressed as a (m + 1) × n matrix.

After obtaining the environmental state space, select the next action based on the policies in the policy network and obtain the next state space. Due to the fact that the tasks of all vehicles are either completed by the vehicles themselves or unloaded to a certain roadbed unit in the model for calculation, the unloading decision of vehicles in the current time slot *t* can be represented as a (m + 1) × 1 dimensional matrix:

$$A_i^t = \left[a_{i,0}^t, a_{i,1}^t, a_{i,2}^t, ..., a_{i,m}^t\right]^T. \quad \textbf{(20)}$$

Due to the indivisibility of vehicle tasks, each task can only correspond to one action, i.e. formula (20) needs to meet the following conditions:

$$\sum_{j=0}^{m} a_{i,j}^t = 1, \;\; i \in (1,2,...,n\}. \quad \textbf{(21)}$$

In summary, the dynamic space of the system can ultimately be represented as a (m + 1) × n dimensional matrix.

From formulas (21), it can be seen that there can be $(m + 1)^n$ action choices in the entire model within a certain time slot. As the number of vehicles and roadbed units in the model increases, the action space will grow exponentially, making it too large and triggering a dimensional explosion. Therefore, multi-agent algorithms were subsequently adopted to solve this problem.

$$A_t = \left[A_1^t, A_2^t, ..., A_n^t\right] = \begin{bmatrix} a_{1,0}^t, & a_{2,0}^t, & a_{3,0}^t, & \cdots, & a_{n,0}^t \\ a_{1,1}^t, & a_{2,1}^t, & a_{3,1}^t, & \cdots, & a_{n,1}^t \\ a_{1,2}^t, & a_{2,2}^t, & a_{3,2}^t, & \cdots, & a_{n,2}^t \\ \vdots, & \vdots, & \vdots, & \cdots, & \vdots \\ a_{1,m}^t, & a_{2,m}^t, & a_{3,m}^t & \cdots, & a_{m,2}^t \end{bmatrix}. \quad \textbf{(22)}$$

Due to the overall goal of the system being to unload and compute vehicle tasks under task constraints while reducing the time cost of completing tasks and maintaining the stability of the virtual energy queue. No matter what action the vehicle takes, the system will receive feedback based on the settings. When a reasonable action is taken, the system will receive a reward after completing the task, defined as:

$$r_k(t) = \alpha \frac{\overline{t} - T_k}{\overline{t}} - \beta E_n(t) E_n. \quad \textbf{(23)}$$

The weight parameters of the reward function are inherited from the parameter values of formula (5-5). If the intelligent agent performs an incorrect action that results in the task not being successfully calculated and returned, the system will be penalized, defined as:

$$PF = \left\{pf_1, pf_2, pf_3\right\}. \quad \textbf{(24)}$$

The three penalty values correspond to the penalties brought by the failure of different task types, and as the importance of the task decreases, the penalty values also gradually decrease.

Therefore, the reward function in this chapter is defined as:

$$r_t = \max\left(\sum_{k=1}^{K} X_k r_k(t) + \sum_{k=1}^{K}(1-X_k)PF_k\right). \qquad \textbf{(25)}$$

Among them, parameters $X_k = 1$ indicates that the task has been successfully processed; $X_k = 0$ indicates that the system has selected an incorrect strategy while processing the task.

According to the description of problem P using formula (5-5), the optimization objective of this chapter is to minimize the time and energy required for the system to complete vehicle tasks. Due to the consideration of task offloading and resource allocation strategies in a trusted environment in this chapter, the trustworthiness of the roadbed unit should also be taken into account, and the trust value should be used as an incentive for the reward function. Therefore, the reward function in this chapter is ultimately defined as:

$$R_t = \max\left(\sum_{i=1}^{n}\left(1+\xi_{i,j}(t)Trust_j(t)\right)\left(\alpha\frac{\overline{t}-T_i}{\overline{t}}+\beta\frac{\overline{e}-E_i}{\overline{e}}\right)+\sum PF\right). \qquad \textbf{(26)}$$

Among them, parameters $\xi_{i,j}(t)$ is a discrete variable that represents the different trust weighting coefficients generated by the system when executing different strategies. When $a_{i,0}^t = 1$, it indicates that the task is calculated locally on the vehicle, which is independent of the trust model, when $\xi_{i,j}(t) = 0$. When the selected roadbed unit for the strategy action $a_{i,j}^t$ is malicious, the parameter $\xi_{i,j}(t)$ is 1, and the reward feedback is reduced by increasing the consumption caused by incorrect strategies; When selecting a normal working roadbed unit, the parameters $\xi_{i,j}(t)$ is -1, which means that the reward feedback obtained at this time is relatively large.

The reward function comprehensively considers the relationship between system time, total energy consumption, and trust. Using trust value as an incentive coefficient can enable the model to quickly identify malicious roadbed units in the system, thereby accelerating training speed and improving system efficiency.

If the original single agent algorithm is directly applied to multi-agent algorithms, each agent will consider other agents as part of the environment, and only their own state experience information will be saved in the experience pool. When the strategies of other agents change, state transitions do not directly reflect in the environment, leading to erroneous guidance, hindering the learning of the agent, and causing algorithm instability. Therefore, in actual multi-agent environments, it is assumed that agent A can obtain the behavioral strategies of other agents and train its own network based on the corresponding experience generated. In the process of inferring strategies through the behavior of other intelligent agents, maximum likelihood estimation can be used to estimate strategies. To avoid overfitting problems caused by policy changes, each agent can learn by training multiple different strategies and balancing the selection of a general strategy.

It can be seen that within a certain time slot in the entire model can have $(m + 1)^n$ kind action choice. As the model of unit in the vehicle number and the increase of the number of action space will be to exponential growth make the space is too big, so as to cause the dimension explosion. Therefore, the multi-agent algorithm is adopted to solve this problem.

If the original single-agent algorithm is directly applied to the multi-agent algorithm, each agent will consider the other agents as part of the environment, and only its own state experience information is stored in the experience pool. When the strategies of other agents change, the state transition will not directly reflect to the environment, which will lead to false guidance, hinder the learning of the agent, and lead to the instability of the algorithm. Therefore, in the actual multi-agent environment, it is assumed that agent A can obtain the behavioral strategies of other agents and train A's own network according to the corresponding experience generated. The maximum likelihood estimation method can be used to estimate the strategy in the process of inferencing the strategy from the behavior of other agents. In order to avoid overfitting problems caused by policy changes, each agent can learn by training several different strategies and choosing a general strategy.

Suppose there are K agents in the centralized training process, and the network parameters are $\theta = \{\theta_1, \theta_2, \ldots, \theta_K\}$. The deterministic strategy of all agents can be expressed as $\mu = \{\mu_{\theta_1}, \mu_{\theta_2}, \ldots, \mu_{\theta_K}\}$. The deterministic policy μ_k for agent k is:

$$\nabla_{\theta_k} J\left(\mu_k\right) = E_{S,A\sim D}\left[\nabla_{\theta_k}\mu_k\left(a_k \left| s_k\right.\right)\nabla_{a_k} Q_k^{\mu}\left(S, a_1, a_2, ..., a_K\right)\Big|_{a_k=\mu_k(S_K)}\right]. \quad \textbf{(27)}$$

The optimization problem is constrained by the following conditions:

$$\alpha_t^m + \sum_{n=1}^{M}\beta_t^{m,n} = 1, \forall m \in M, \forall t \in T. \quad \textbf{(28)}$$

Many methods in reinforcement learning utilize the recursive relationship between state action functions, known as the Bellman equation. Under the optimal strategy, the Bellman optimal equation for the state action function can be written as:

$$Q^*(s,a) = E\left[R(s,a) + \gamma \max_{a'} Q^*(s',a')\right]. \quad \textbf{(29)}$$

Among them, s' represents the next state that transitions from state s to under action a.

For Cirtic networks, updates can be made according to the loss function:

$$Loss(\theta_k) = E_{S,A,S',R}\left[\left(Q_k^{\mu}\left(S, a_1, a_2, ..., a_K\right) - y\right)^2\right]. \quad \textbf{(30)}$$

$$y = r_k + \quad Q_k^{u'}\left(S', a_1', a_2', ..., a_K'\right)\Big|_{a_j' = \quad '_j\left(s_j\right)}. \quad \textbf{(31)}$$

The Actor network can be updated by minimizing the policy gradient of the agent:

$$\nabla_{\theta_k} J \approx \frac{1}{Z}\sum_j \nabla_{\theta_k}\mu_k\left(s_k^j\right)\nabla_{a_k} Q_k^{\mu}\left(S^j, a_1^j, a_2^j, ..., a_K^j\right)\Big|_{a_k=\mu_k\left(s_k^j\right)}. \quad \textbf{(32)}$$

In order to ensure the stability of the training process, DDPG algorithm uses "soft update" to update some parameters of the target network. The update relation is:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + \left(1-\tau\right)\theta^{Q'}. \quad \textbf{(33)}$$

It is difficult to obtain accurate solutions for reinforcement learning problems with high-dimensional state and action spaces by directly maximizing the Q function. Use deep reinforcement learning to obtain approximate solutions to solve this problem. It can be seen from the previous section that the action space contains the computing resources provided by the edge computing server ESC for each intelligent device $f_{ESC,t}$, the allocation of computing resources is continuous rather than discrete in practical scenarios, so the DDPG algorithm is chosen in deep reinforcement learning algorithms. DDPG is a deep deterministic policy gradient algorithm proposed to solve continuous action control problems.

In DDPG, two independent deep learning networks are used, where the critical network $Q(s, a|\theta^Q)$ Approximate Q-function, actor network $\mu(s|\theta^\mu)$ Approximation strategy function. Among them, θ^Q and θ^μ The neural network weights for the critical network and the actor network, respectively. In addition, $Q'(s, a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ Representing the target critical network and target actor network respectively. The standard network is used to calculate the target value, in order to $\theta^{Q'}$ and $\theta^{\mu'}$. For its corresponding neural network weights. The target network is a lagged copy of its original network, which replicates the learning network every certain number of steps to improve learning stability.

This chapter combines the DDPG algorithm with the actual scenario of vehicular intersection and proposes the TBMADDPG algorithm. The algorithm process is shown in Table 1.

Table 1. TBMADDPG training algorithm description for N agents

```
Algorithm 1
1    Randomly initialize all Actor and Critic networks and their respective weight parameters and their corresponding experience pools
2    Initializes information such as computing resources and network status in the system model
3    Initializing the direct and global trust matrix, and randomly specifying the malicious subgrade unit index
4    For episode = 1 to max_ episodes:
5      Initialize the action in the process of exploring the environment parameters: noise variable $N_t$ subgrade unit, vehicle and information;
6      For time = 1 to max_ slots:
7      Randomly generate and sort vehicle task information in the current time slot.
8        Update the subgrade unit resource allocation in the model
9        For task = 1 to max_ indexes:
10         Each agent outputs actions based on the current policy network and noise perturbations and constraints $a_i = \mu_{\theta_i} + N_t$
11         perform the action $a = (a_1, a_2, \ldots, a_n)$ to get rewarded $r$, and the next status $s'$
12         Waiting time delay queue updates, and the sample data $(s, a, r, s')$ deposit pool $R$ experience
13       End For
14       Calculate the global trust of all vehicles to the subgrade unit under the current time slot and update it
15       For agent = 1 to $N$:
16         Randomly sample $Z$ bars of data $(s^j, a^j, r^j, s'^j)$ in experience pool $R$ to form mini batch
17         Update the online network parameters of Critic by formula (1).
18         Update the online network parameters of Actor by formula (2).
19       End For
20       Update the Target network parameters of each agent by formula (3)
21     End For
22   End For
```

4 Simulation and Result Analysis

In order to verify the validity of resource allocation strategy based on trust model in vehicle edge computing network, subgrade units are divided into normal subgrade units and malicious subgrade units. The malicious subunit not only provides false computing resources but also leaks task information, so the task should be avoided to be unloaded into the malicious subunit. In the simulation, the model randomly selects two subgrade units as malicious subgrade units. Regardless of the type of subgrade unit, all vehicles have an initial trust rating of 0.5 in them.

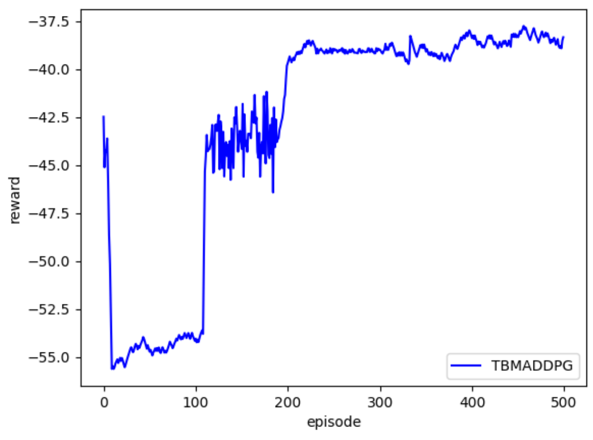

Fig. 1. Algorithm convergence curve

Fig. 1 shows the training results of the TBMADDPG algorithm. As can be seen from the figure, after several fluctuations, the algorithm reaches a stable state in 400 rounds. The effect of the model increases rapidly in about 100 rounds because the multi-agent model first stores the experience group into its own experience pool during interactive learning, and then trains the network when the capacity of the experience pool is full. In fact, the model has achieved a good training effect in about 250 rounds, so the training speed of TBMADDPG algorithm is still relatively fast, and a reasonable unloading scheme with minimum delay and energy consumption based on the trust model can be obtained.

5 Future Work

According to the research in this paper, the following are some existing problems and the summary and prospect of future work.

In the simulation of this paper, the proposed algorithms are all based on DDPG algorithm. Considering the short-comings of DDPG algorithm itself, other reinforcement learning algorithms, such as PPO and A3C, can be adopted in the subsequent research for simulation solutions, which may get better training results or achieve the purpose of simplifying the work.

The actions taken by the agent in this paper are actually from the direct interaction between the vehicle and the subgrade unit, and there is a "heavy unloading" situation in real life, that is, some actions need to be judged by a third party whether to continue to perform. In the face of this problem, the current reinforcement learning is difficult to learn the optimal strategy because of the selfishness of the third-party system nodes. In order to solve this problem, we can consider the integration of the models in Chapter 3 and Chapter 4, and nest the trust degree as the selfless incentive in addition to the incentive of the income function. In this way, it may be helpful for reinforcement learning to solve practical problems like "heavy unloading".

6 Acknowledgement

This research was funded by local scientific and technological development guided by the central government 226Z0306G and National Natural Science Foundation of China W21A0300060.

References

[1] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of machine learning, second ed., London, England, 2018 (Chapter 6).

[2] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, M. Imran, The Role of Edge Computing in Internet of Things, Communications Magazine 56(11)(2018) 110–115.

[3] B. Manickavasagam, B. Amutha, M. Revathi, N. Karthick, K. Sree, K. Priyanka, Wireless body area network mutual trust analysis technique for fault detection using software defined network (SDN), Journal of Intelligent & Fuzzy Systems 40(1)(2021) 575–589.

[4] N. Paul, D. Raj, Enhanced trust based access control for multi-cloud environment, CMC-Computers Materials & Continua 69(3)(2021) 3079–3093.

[5] H. Hu, Y. Han, M. Yao, X. Song, Trust based secure and energy efficient routing protocol for wireless sensor networks, IEEE Access 10(1)(2021) 10585–10596.

[6] Y. Zhu, Y. Wang, Malicious vehicle node detection mechanism based on repeated game and trust evaluation in VANET, in: Proc. The 2020 3rd International Conference on Computer Information Science and Application Technology (CISAT), 2020.

[7] Y. Ouyang, W. Liu, Q. Yang, X. Mao, F. Li, Trust based task offloading scheme in UAV-enhanced edge computing network, Peer-to-Peer Networking and Applications 14(1)(2021) 3268–3290.

[8] A. Rago, G. Piro, G. Boggia, P. Dini, Anticipatory allocation of communication and computational resources at the edge using spatio-temporal dynamics of mobile users, IEEE Transactions on Network and Service Management 18(4)(2021) 4548–4562.

[9] C. Zheng, S. Liu, Y. Huang, L. Yang, Hybrid policy learning for energy-latency tradeoff in mec-assisted vr video service, IEEE Transactions on Vehicular Technology 70(9)(2021) 9006–9021.

[10] J. Wang, L. Zhao, J. Liu, N. Kato, Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement

Learning Approach, IEEE Transactions on Emerging Topics in Computing 9(3)(2021) 1529–1541.

[11] H. Zhang, J. Liu, H. Zhao, P. Wang, N. Kato, Blockchain-Based Trust Management for Internet of Vehicles, IEEE Transactions on Emerging Topics in Computing 9(3)(2021) 1397–1409.

[12] J. Zhou, F. Wu, K. Zhang, Y. Mao, S. Leng, Joint optimization of offloading and resource allocation in vehicular networks with mobile edge computing, in: Proc. 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), 2018.

[13] Y. Liu, S. Wang, J. Huang, F. Yang, A computation offloading algorithm based on game theory for vehicular edge networks, in: Proc. IEEE International Conference on Communications (ICC), 2018.

[14] J. Du, F. Yu, X. Chu, J. Feng, G. Lu, Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization, IEEE Transactions on Vehicular Technology 68(2)(2019) 1079–1092.

[15] W. Tan, W. Lau, O. Yue, T. Hui, Analytical models and performance evaluation of drive-thru internet systems, IEEE Journal on selected areas in Communications 29(1)(2011) 207–222.

[16] Y. Hui, Z. Su, T. Luan, J. Cai, Content in motion: An edge computing based relay scheme for content dissemination in urban vehicular networks, IEEE Transactions on Intelligent Transportation Systems 20(8)(2019) 3115–3128.

[17] Z. Su, Y. Hui, T. Luan, Distributed task allocation to enable collaborative autonomous driving with network softwarization, IEEE Journal on Selected Areas in Communications 36(10)(2018) 2175–2189.

[18] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, Y. Qin, Joint optimization of computation offloading and task scheduling in vehicular edge computing networks, IEEE Access 8(1)(2020) 10466–10477.

[19] C. Lai, W. Chien, L. Yang, W. Qiang, LSTM and Edge Computing for Big Data Feature Recognition of Industrial Electrical Equipment, IEEE Transactions on Industrial Informatics 15(4)(2019) 2469–2477.

[20] Q. Xu, Z. Su, Y. Wang, M. Dai, A trustworthy content caching and bandwidth allocation scheme with edge computing for smart campus, IEEE Access 6(1)(2018) 63868–63879.

[21] Y. He, C. Liang, F. Yu, Z. Han, Trust-based social networks with computing, caching and communications: A deep reinforcement learning approach, IEEE Transactions on Network Science and Engineering 7(1)(2020) 66–79.

[22] M. Alam, A. Jamalipour, Multi-Agent DRL-Based Hungarian Algorithm (MADRLHA) for Task Offloading in Multi-Access Edge Computing Internet of Vehicles (IoVs), IEEE Transactions on Wireless Communications 21(9)(2022) 7641–7652.

[23] W. Duan, J. Gu, M. Wen, G. Zhang, Y. Ji, S. Mumtaz, Emerging Technologies for 5G-IoV Networks: Applications, Trends and Opportunities, IEEE Network 34(5)(2020) 283–289.