Journal of Computers Vol. 36 No. 2, April 2025, pp. 119-137
https://doi.org/10.63367/19911599202504360200 9

Resource Management of Connected Vehicles Based on Trust Model in Intersection Scene

Haocheng Lang*, Zhenjiang Zhang, Zhaonian Li, Qingyu Zhao, and Qianli Yang

School of Electronic Information Engineering, Beijing Jiaotong University,
Beijing 100044, China
22211145@bjtu.edu.cn, zhangzhenjiang@bjtu.edu.cn, 20120083@bjtu.edu.cn,
24120157@bjtu.edu.cn, 24120134@bjtu.edu.cn

Received 29 March 2025; Revised 13 April 2025; Accepted 16 April 2025

Abstract. This paper focuses on the resource management and trust issues in the context of vehicular networks at intersections, proposing a trust model-based resource management scheme for the Internet of Vehicles. A lightweight trust evaluation model is constructed, which dynamically assesses the trust relationship between vehicles and roadside units through three dimensions: task processing timeliness, task completion satisfaction, and resource allocation rate. Simulation results demonstrate that the model can effectively identify malicious roadside units, reduce system time and energy consumption, and enhance overall system performance.

Keywords: vehicular networks, direct trust model, resource allocation

1 Introduction

To explore the challenges of trust and resource management for interactions between vehicles and roadside infrastructure at urban intersections [1], this study introduces a streamlined framework for trust assessment. Within this framework, vehicles evaluate the reliability of roadside units by integrating both direct and indirect trust metrics [2]. This approach facilitates secure and efficient task processing or offloading for vehicles within the network [3]. The model's effectiveness and robustness are validated through comparative simulations against established methods [4].

The rest of this paper is structured as follows: second section presents the extant research. The third section, details the system model of trust-based resource management in intersection scenarios. The development of a direct trust model is outlined in forth section. The fifth section presents the conclusions of the study and proposes future research directions.

2 Related Work

In recent years, edge computing has become a hot topic in the academic community. After putting forward the basic definition of edge computing, Shi et al. Instantiated its actual application scenarios, such as cloud edge collaboration, resource scheduling, smart city, etc. [5]. As a hot topic in the field of edge computing, task offloading and resource scheduling refer to the technology of choosing to offload the generated tasks to the edge server with more abundant resources for processing in the scenario of limited computing and energy resources of edge devices. With the deepening of research, from the single research of edge computing scenario, more and more literatures are devoted to exploring the optimal task allocation strategy between vehicles and edge computing servers in the Internet of vehicles scenario, which makes the research more practical.

In order to make decisions at the overall level, Zhang et al. Introduced a task unloading architecture based on Software Defined Network (SDN) to collect information in the entire vehicle network [6, 7]. Affected by previous research under mobile edge computing (MEC), vehicle edge computing (VEC) deploys edge servers

* Corresponding Author

in roadside units to provide additional computing and storage capacity for the overall network. In order to make the VEC model more realistic, Huang et al. Classified the tasks generated in the Internet of vehicles according to different needs. Task offloading in edge networks is closely related to resource allocation. Most studies focus on the optimization of offloading strategy with the goal of minimizing delay and energy consumption or a compromise between the two. In recent years, many researchers [8-14] have focused on the optimization of unloading strategy under VEC model, and have made a series of significant contributions in reducing time consumption and encouraging cooperation between vehicles.

For the direction and path of task unloading, a new communication mode is proposed in research [15]: the task vehicle sends the task to its moving direction to the edge server through the multi hop V2V communication mode. In this way, the cooperation between vehicles can be encouraged and the coverage of the overall network can be expanded. Khayyat et al. Chose a two-dimensional unloading direction in their research, and the task was calculated locally or unloaded to the edge node for processing [14]. In combination with V2V and V2I, Zhao and others believe that in addition to the above two flow directions, tasks can also reward other vehicles that assist in completing tasks, making full use of all computing resources in the vehicle network [16]. In addition, the system will give priority to allocating computing resources to vehicles that have won awards. In addition to the research on the optimal strategy in typical urban scenarios, some researchers [17-19] also carried out a series of research in the vehicle edge computing environment assisted by unmanned aerial vehicle (UAV), and realized the information interaction task in remote network scenarios with the help of satellite and UAV network. This scenario is also one of the popular scenarios in the current research on resource offload.

The growing adoption of machine learning, driven by its transformative capabilities in Internet of Things applications, has led researchers to increasingly leverage this technology for optimizing task offloading and resource allocation strategies. For instance, studies [20, 21] employ the Long Short-Term Memory (LSTM) algorithm to forecast edge computing and communication resources for mobile devices, enabling informed decision-making based on anticipated future trends. Regarding optimization simulations, Wang et al. [22] developed a sophisticated resource distribution framework using a Deep Q-Network (DQN), tailored for dynamic edge computing environments, which effectively minimizes latency. However, DQN-based approaches, including standard DQN and Double DQN, often incur high computational costs when addressing multiple discrete offloading decisions. Consequently, policy-driven techniques, such as Actor-Critic Deep Reinforcement Learning (AC-DRL) and Deep Deterministic Policy Gradient (DDPG), have gained traction in recent investigations for their efficiency. Moreover, to tackle the complexities of computation-intensive vehicular applications, multi-agent reinforcement learning has been adopted to stabilize environmental dynamics and ensure consistent policy updates [23].

3 System Architectures

This chapter discusses the problem of vehicle task unloading and scheduling in a common scenario in the vehicle network, that is, how to ensure the optimal unloading of vehicle tasks in the Urban Central Road intersection scenario. As shown in Fig. 1, it is assumed that there are M subgrade service units including normal and malicious types at the intersection, which are connected with each other through optical fibers and are represented by sets $M = \{1, 2, \cdots, m\}$. The normal subgrade unit can provide computing resources and safe processing environment to assist the vehicle to complete the unloading calculation of the task; Erroneous service information and the unauthorized release of a vehicle's private data can stem from a compromised subgrade unit. In the model discussed in this chapter, the calculation resources and usage of subgrade element are dynamic. Because the vehicles at the road intersection are very dense, each vehicle in the figure is actually an abstract simulation of multiple vehicles. For example, the multi vehicle structure in the lower left corner of the model figure is a concrete display. This abstract vehicle model can be represented by sets $N = \{1, 2, \cdots, n\}$. The vehicle mentioned later in this chapter refers to this multi vehicle collection in addition to its own meaning.

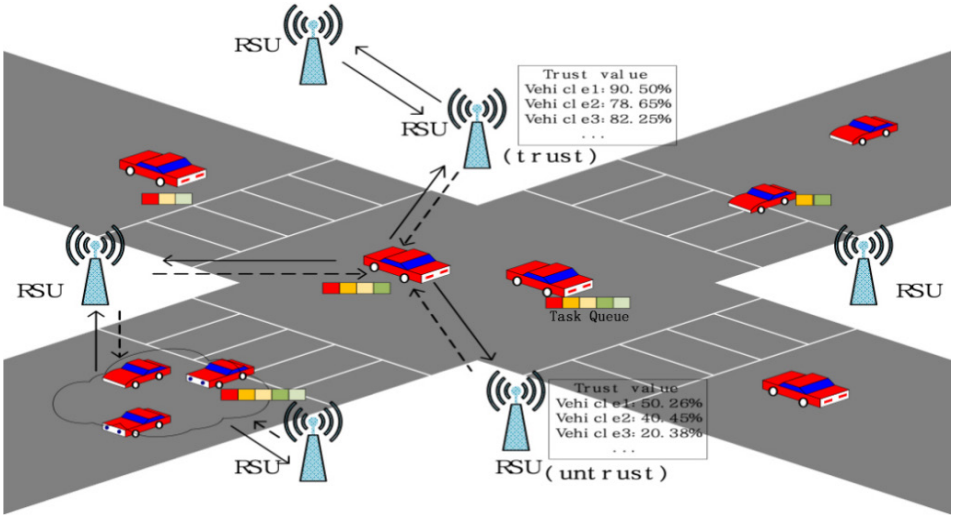

Fig. 1. Vehicle edge computing network model at intersection

When the vehicle determines to unload the generated task, the task will be migrated to the subgrade processing unit through the wireless link, and then the task will be processed by the allocated resources. After considering the safety and security of connected automobiles, the model needs to introduce a reliable trust evaluation system. The vehicle and the subgrade unit engage in communication during every defined time interval, the trust value is evaluated and updated, and then the malicious service behavior in the system is effectively detected. The model of task unloading and resource allocation process in a trusted environment can be shown in Fig. 2:

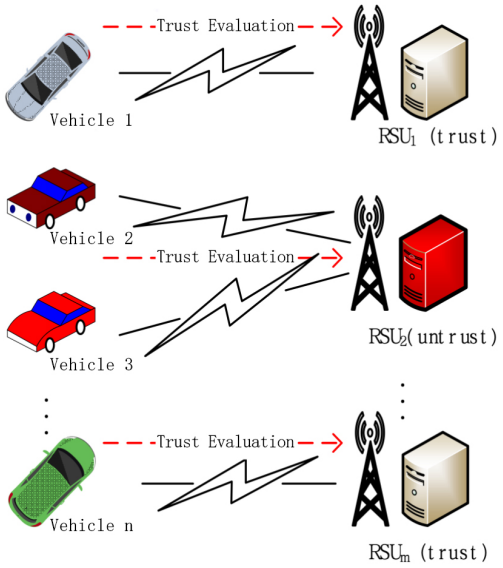

Fig. 2. Trusted task offloading model of multiple vehicles and multiple roadside units

In this paper, a trust model based task unloading and resource scheduling model is established for vehicle edge computing scenarios. The dynamic trust value is used as an incentive to enable vehicles to quickly distinguish malicious subgrade units from normal subgrade units. In addition, due to the large number of vehicles in the model, the single agent algorithm is faced with problems such as action space explosion and difficult convergence. In the final simulation stage, the multi-agent distributed deep reinforcement learning model is used to solve the problem. According to the traffic scene at the intersection, this chapter introduces TBMADDPG, a distributed algorithm designed for resource scheduling, which is founded on a dependable model and deep reinforcement learning principles, and proves the reliability and effectiveness of the algorithm in the following simulation comparison.

3.1 Vehicle Speed Model

Based on the research of Shi et al. And Tan et al. [14, 24, 25], this paper introduces a model for free traffic flow designed to characterize vehicle speed data. Within a defined cell, it is assumed that all vehicles maintain a uniform speed, and this speed adheres to a Gaussian distribution. Consequently, the relationship between traffic density and mean vehicle velocity is expressed as:

$$\overline{v} = v_{\max}\left(1-\frac{\rho}{\rho_{\max}}\right) \tag{1}$$

Where ρ is the traffic density inside the subgrade unit, $v_{\max}$ and $\rho_{\max}$ are the maximum values of speed and traffic density. Each vehicle initializes its speed independently and randomly in each time period, which is recorded as $v \sim F(\overline{v}, \sigma_v^2)$. In the actual road situation, the higher the vehicle density, the smaller the speed difference between vehicles in order to reduce the probability of vehicle collision. Therefore, similar to the research of Su et al. [26], this paper sets the variance of vehicle speed and average speed as a positive correlation, which is recorded as $\sigma_v \propto \overline{v}$.

3.2 Vehicle Task Model

This chapter sets that each vehicle generates and executes at least one task in each time slot, and all vehicle tasks are in the complete migration mode, that is, the tasks are either calculated by the vehicle itself or all migrated to the subgrade unit for processing. In each time slot, vehicle *i* generates multiple types of tasks, expressed as $Y = (C, D, \Phi)$, where C is the number of CPU revolutions required to complete the task, D is the size of the input task data, different types of vehicle tasks have different delay tolerance, and $\tau_{\max}^{i}$ is used to represent the maximum acceptable delay of type *i* task. Φ represents the type of vehicle task, which has three types:

1) φ_1 represents the computing tasks related to security, that is, the tasks with the highest priority. These tasks have the highest requirements for time delay and are all calculated locally in the vehicle. The maximum delay threshold is set to $\tau_{\max}^{1}$.

2) φ_2 represents some computing tasks related to assisted driving, such as vehicle navigation, optional safety applications, etc. these tasks are defined as high priority tasks, and the vehicle can select local processing or unloading to its subgrade processing unit.

3) φ_3 task represents the low priority task related to entertainment generated by the vehicle, and all such tasks are handed over to the subgrade unit for calculation and processing.

Since there are only two processing paths for tasks in the model set in this chapter, that is, each vehicle task can only be calculated locally or unloaded to a nearby subgrade unit RSU_j for processing, the task unloading decision of vehicle *i* in a certain time slot can be expressed as a vector of m+1 dimension:

$$A_t^i = [a_t^{i,0}, a_t^{i,1}, a_t^{i,2}, \cdots, a_t^{i,m}] \tag{2}$$

Where $a_t^{i,0}=1$ indicates that the task is not unloaded and the calculation is completed locally; $a_t^{i,j}=1$ indicates that the task of vehicle *i* is transferred to Subgrade unit RSU_j for processing. Since the task is indivisible and subject to the dual unloading strategy, the unloading strategy of the task at each time must meet the following conditions:

$$\sum_{j=0}^{m} a_t^{i,j}=1, i\in\{1,2,\cdots,n\} \tag{3}$$

Finally, the unloading decision of all vehicles in the model can be represented by a matrix of $i\times j$

$$A_t=[A_t^1,A_t^2,\cdots,A_t^n]=\begin{bmatrix} a_t^{1,0},a_t^{2,0},\cdots,a_t^{n,0} \\ a_t^{1,1},a_t^{2,1},\cdots,a_t^{n,1} \\ \vdots\ ,\ \vdots\ ,\cdots,\ \vdots \\ a_t^{1,m},a_t^{2,m},\cdots,a_t^{n,m} \end{bmatrix} \tag{4}$$

3.3 Communication Model

Tasks originating from vehicles are processed and offloaded using diverse methods within the vehicular edge computing system, influenced by factors such as latency, power usage, channel conditions, and vehicle speed. The offloading decisions, represented by the set $X\in\{0,1,2\}$, dictate the task allocation behavior across zones managed by roadside units, formalized as $\{a_1,a_2,\ldots,a_n\}$. Within this decision framework, a value of 0 denotes local computation of the task on the vehicle, 1 indicates offloading to the vehicle's designated roadside unit, and 2 signifies collaboration with a nearby roadside unit for task execution.

Within the SDN-VN framework presented in this chapter, the adoption of massive MIMO technology enables roadside units to communicate simultaneously with several vehicles [26]. To streamline the model and ease computations, the bandwidth of communication channels between various vehicles and roadside units is assumed to be uniform, denoted by the parameter ω. Additionally, a value of $x=1$ signifies that a task has been offloaded, whereas $x=0$ denotes that the task remains unloaded.

In a real-world scenario, the condition of the communication link is also susceptible to noise. Based on the Shannon formula, the information transfer rate within the wireless communication channel can be determined as follows:

$$R_n=w\cdot\log_2\left(1+\frac{p_n h_n}{\sigma^2+I_n^{RSU}}\right), for\ n\in N \tag{5}$$

Where p_n is the uplink power between vehicle *n* and subgrade unit, h_n is the uplink gain, σ^2 is the noise energy, and I_n^{RSU} is the interference in the channel band. When the vehicle n unloads the task to the subgrade processing unit, the uplink power is limited to: $P=\{p_n|0\le p_n\le \mathrm{P}_n\}$. When $p_n=0$, it means that the task will not be unloaded to the edge side for calculation.

3.4 Computation Model

For tasks of type φ_1, the vehicle's local computation model is the only factor that needs consideration. Every vehicle is equipped with a central processing unit (CPU) to process or compute the tasks it generates. In order to process task Y_n generated by computing, the computing resources allocated by each vehicle for computing tasks

need to comply with the allocation restriction policy $F = \left\{ f_n^{loc} \mid 0 < f_n^{loc} \le F_n^{loc}, n \in N \right\}$. In addition, since type φ_1 tasks are only calculated locally in the vehicle, and the whole process does not involve information transmission and other behaviors, the local execution time and energy consumption of tasks can be expressed as:

$$t_1 = \frac{C_n}{f^{loc}} \tag{6}$$

$$E_1 = \kappa \left(f_n^{loc} \right)^2 C_n \tag{7}$$

The parameter k, which represents the power coefficient, has a value determined solely by the CPU's architectural design.

For tasks categorized under type φ_2, two computational approaches must be evaluated: local processing within the vehicle or offloading to the associated roadside unit (RSU) for execution. Local computation in the vehicle can proceed using the previously outlined model. In contrast, the offloading approach requires accounting for the task's upload and download phases; however, given the typically small size of the task data, the feedback results are deemed negligible. Consequently, the energy and time costs associated with feedback can be disregarded. Thus, the overall time required for offloading and processing a φ_2 type task encompasses both the computation duration and the upload time at the roadside unit, defining the total time consumption for such tasks as follows:

$$t_2 = \begin{cases} t_2^{loc} = C_n \Big/ f_n^{loc} \\ t_2^{RSU} = t_2^{exu} + t_2^{up} = C_n \Big/ f^{RSU} + D_n \Big/ R_n \end{cases} \tag{8}$$

For an offloading task, the energy expenditure that must be evaluated is limited to the power used by the vehicle during the upload phase; thus, the total energy required by the vehicle to execute a task of type φ_2 is expressed as follows:

$$E_2 = \begin{cases} E_2^{loc} = \kappa \left(f_n^{loc} \right)^2 C_n \\ E_2^{RSU} = p_n^k \dfrac{D_n}{R_n} \end{cases} \tag{9}$$

For tasks classified as type φ_3, two computational approaches must be assessed: local processing within the vehicle or offloading to the associated roadside unit (RSU) for execution. However, due to the distinct nature of this task type, the resulting data from computation cannot be overlooked, with its size being η times larger than the original task size. Furthermore, as data transfer between adjacent roadside units occurs via optical fiber, the associated transmission latency is considered negligible in this study. Based on the described model, the equations for calculating the time and energy requirements of φ_3 type tasks are presented as follows:

$$t_3 = \begin{cases} t_3^{RSU} = t_3^{exu} + t_3^{up} + t_3^{down} = C_n \Big/ f^{RSU} + (1+\eta) D_n \Big/ R_n \\ t_3^{adj\_RSU} = C_n \Big/ f^{adj\_RSU} + (1+\eta) D_n \Big/ R_n \end{cases} \tag{10}$$

$$E_3 = E_3^{RSU} = E_3^{adj\_RSU} = (1+\eta)\, p_n^k \frac{D_n}{R_n} \tag{11}$$

When calculating the task generated by the vehicle itself, the task unloading strategy is $a_t^{i,0} = 1$, and the temporal and energetic costs incurred during this period can be determined by employing equations (10) and (11).

When the unloading strategy is $a_t^{i,j} = 1$ and $j \neq 0$, the task selection is migrated to the subgrade unit RSU_j for processing. In this chapter, the task processing queue is set to conform to FIFO mode, that is, the task generated first in each time slot receives service first. Different subgrade units have different computing resources, which are shared by multiple vehicles, and the task number and arrival order of vehicles are uncertain. In this chapter, it is set that the resources in the subgrade unit are fixed, but the computing resources provided to the task vehicle in each time slot are random and dynamic, and $Res_{i,j}(t)$ represents the amount of resources allocated by vehicle *i* and subgrade unit RSU_j under time slot *t*. In addition, for the convenience of calculation, the task return step is not considered in this chapter. Therefore, according to the analysis and discussion in Chapter 3, The duration and energy expenditure of the vehicle task when offloaded to the roadside unit are defined as follows:

$$t_2 = t_2^{exu} + t_2^{up} = \frac{C_n}{Res_{i,j}} + \frac{D_n}{R_n} \tag{12}$$

$$E_2 = p_n^k \frac{D_n}{R_n} \tag{13}$$

In the time delay calculation model, the waiting time delay shall also be added to the total time delay for completing vehicle tasks, which is expressed as:

$$T = t_i + t_w \tag{14}$$

4 Direct Trust Model

4.1 Task Processing Timeliness Model

In the Internet of vehicles, the subgrade unit is responsible for the functions of message forwarding and task execution. If it takes a long time to complete various tasks generated by the vehicle, the timeliness of message transmission will be reduced. When facing some delay sensitive tasks, the reduction of timeliness also means the improvement of the error of message delivery. Therefore, define the timeliness function of the current task processing, as shown in formula (15):

$$t_{time} = \begin{cases} 1 - \dfrac{t_{cur} - t_{exist}}{\tau_{\max}} & t_{cur} - t_{exist} \le \tau_{\max} \\ 0 & otherwise \end{cases} \tag{15}$$

t_{cur} represents the current time, t_{exist} represents the time when the current task is generated, and $\tau_{\max}$ represents the longest latency experienced by the task. In formula (15), the above three factors are used to measure the timeliness of task processing. The greater the value of t_{time}, the more timely the task processing.

After the vehicle and subgrade unit nodes successfully interact with each other for one time, the system will generate a loss of timeliness, which is defined as $loss_{time} = \alpha_i \times (1 - t_{time})$, where α_i means that the task of vehicle *i* is successfully processed. Furthermore, the timeliness loss rate of processing tasks can be defined as formula

(16):

$$t_{loss} = \alpha_i \times (1 - t_{time}) \Big/ \alpha_i = (1 - t_{time}) \tag{16}$$

According to the above timeliness loss rate and urgency of task processing, use formula (16) to describe the timeliness trust of task processing:

$$T_{i,j}^{\varphi} = (1 - t_{loss})^{e^{\varphi - \min\{\phi\}} \cdot t_{loss}} \tag{17}$$

Where min function represents the minimum value of the parameter set.

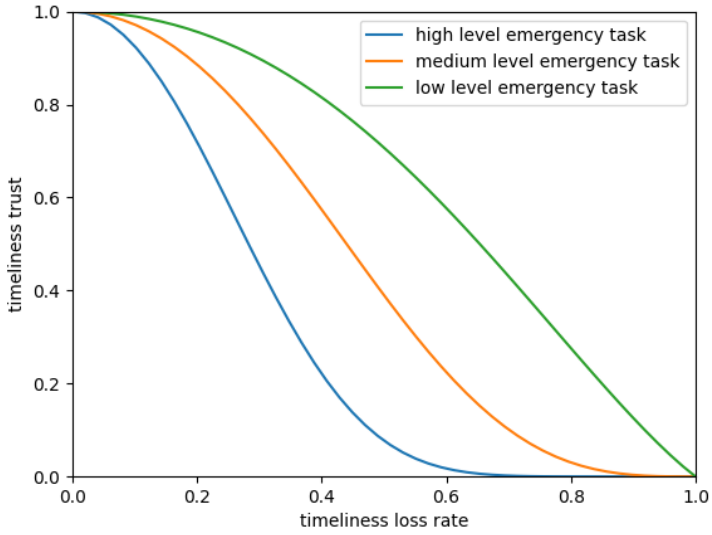

Fig. 3. The relationship between timeliness loss rate and timeliness trust

According to the definition and the curve in the Fig. 3, when the timeliness loss rate t_{loss} of tasks is 0, the timeliness trust of all types of tasks is 1; When the timeliness loss rate t_{loss} of tasks is 1, the timeliness trust of all types of tasks is 0. The trust function also obtains different trust values according to different task types. When the loss rate of timeliness is the same, the trust degree of timeliness brought by tasks with high urgency is relatively low. Because tasks with high urgency are more important than low-level tasks, once a task fails, the system will pay a higher price, and the corresponding trust value should also be the lowest. In formula (4-3), the task urgency is indexed to better distinguish the timely trust curves of different types of tasks. Compared with linear processing, it can bring more punishment when the task with high urgency fails.

Weighted average the timeliness trust of all information over a period of time to obtain the final timeliness trust value:

$$T_{i,j}^{t} = \sum \varphi \cdot T_{i,j}^{\varphi} \Big/ \sum_{\Delta t} \alpha_i \tag{18}$$

4.2 Task Completion Satisfaction Model

When the vehicle chooses to unload the task to the subgrade unit for interaction, the quality of the data interaction needs to be evaluated. If the task can be effectively executed and finished it indicates that the reliability of the subgrade unit is high and should be highly evaluated; If the task interaction fails, the reverse is true. According to the historical interaction data between the two nodes of vehicle *i* and subgrade unit RSU_j, the task completion satisfaction between nodes is calculated by using the evaluation method based on Bayesian trust [26]. Set parameters $s_{i,j}$ and $f_{i,j}$ to indicate the number of successful and failed interactions between the vehicle and the edge node. Over a specified duration, the trust value of this task completion satisfaction is defined as:

$$T_{sf}^{t} = \mathbb{E}\left(Beta\left(s_{i,j}+1, f_{i,j}+1\right)\right) = \frac{s_{i,j}+1}{s_{i,j}+f_{i,j}+2} \tag{19}$$

In the above formula, the satisfaction of the task is analyzed from the historical interaction data between nodes and the trust degree for the future is analyzed. However, this trust is dynamic and will change with time. In order to ensure the timeliness of trust evaluation between nodes, this chapter introduces the parameter of time attenuation penalty factor to evaluate the timeliness of time-related trust model. The dynamic update of trust between nodes should meet the conditions the longer the current time, the smaller the impact of historical interaction data on trust evaluation in the current state. Therefore, the time attenuation penalty factor is defined as shown in formula (20):

$$pen(\Delta t_0) = \frac{1}{1+e^{-(t-t_0)}} = \frac{1}{1+e^{-\Delta t_0}} \tag{20}$$

When the behavior of a node shows abnormal or malicious behavior, the time decay factor will quickly reduce the trust value of the node. When the behavior of the node is improved or normal, trust in the node will slowly recover. Compared with the linear attenuation factor, the above formula can find abnormal nodes by reducing the trust value of abnormal nodes more quickly, and is also more cautious about restoring trust to untrusted nodes.

4.3 Resource Allocation Rate Model

In the interaction between vehicle and subgrade processing unit, the amount of computing resources provided by subgrade unit will also directly affect the trust between vehicle and subgrade unit. Set $Res_{i,j}(t)$ as the number of computing resources that subgrade unit RSU_j can provide for vehicle *i* in time slot *t*, and $Res_j(t)$ as the total amount of computing resources that subgrade unit RSU_j can provide in time slot *t*, then its resource allocation rate can be expressed as:

$$T_R^t = \frac{Res_{i,j}(t)}{Res_j(t)} \tag{21}$$

Due to frequent communication connecting the vehicle with the roadside infrastructure unit, and the value of the interaction experience far away from the current time for the iterative update of the current trust value is not great, the interaction information occurred recently should be taken into account when calculating the direct trust degree, and some long-distance experience should be abandoned to a certain extent. Therefore, the idea of sliding window can be used to update the direct trust, as shown in Fig. 4. Whenever the interaction experience of a new time interval begins, the oldest record within the observation window is removed, and only the valid interaction data within that window is evaluated. At the same time, this can also reduce the calculation cost of trust.

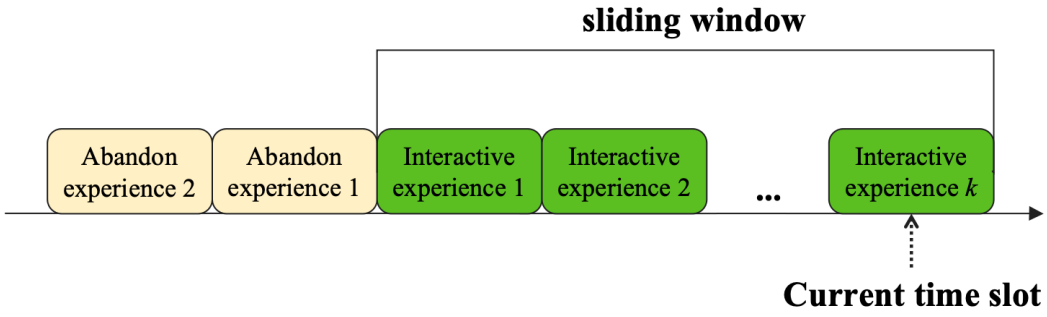

Fig. 4. Sliding window

Similar to the method of calculating direct trust in reference [27], the direct trust between vehicle *i* and subgrade unit RSU_j in time slot *t* can be expressed by formula (22):

$$DTrust_{i,j}(t) = \frac{\sum_{k=1}^{K}(T_{i,j}^k + T_{sf}^k + T_R^k)}{\sum_{k=1}^{K}(T_{i,j}^k + T_{sf}^k + T_R^k) + \sum_{k=1}^{K} pen(\Delta t_k)(2 - T_{i,j}^k - T_{sf}^k)} \tag{22}$$

Where, the parameter *K* represents the amount of experience used in the window.

5 Indirect Trust Model and Global Model

5.1 Indirect Trust Model

In addition to direct trust, indirect observations from other agents additionally contribute significantly in the evaluation of trust. The introduction of indirect trust model can help to find the deceptive behavior of the observed subgrade processing unit only to some or a certain user, and avoid the subjective bias of evaluating the trust of edge nodes only derived from past data experience. Therefore, collecting and combining other vehicle nodes for indirect trust evaluation of the subgrade treatment unit may enhance the dependability and precision of the overall trust evaluation process. As an effective reasoning mechanism to deal with uncertainty, Dempster Shafer evidence theory can effectively and reasonably fuse the data of multi-user evidence elements.

In the indirect trust analysis, this chapter considers that the trust evaluation of different vehicles on different subgrade treatment units is independent of each other, and takes the direct trust between the two as the basic probability distribution, that is, the quality function of DS is defined as:

$$m_i\left(H_j\right) = DTrust_{i,j}(t) \tag{23}$$

$m_i\left(H_j\right)$ represents the trusted value that vehicle *i* considers subgrade processing unit RSU_j as a normal working node in this time slot. According to Dempster's composition rule of finite mass functions, the secondary trust level for vehicle *i* to Subgrade processing unit RSU_j in time slot *t* is defined as:

$$\begin{aligned} ITrust_{i,j}(t) &= \left(m_1 \oplus m_2 \oplus \cdots \oplus m_n\right)(H) \\ &= \frac{1}{K} \sum_{H_1 \cap H_2 \cap \cdots \cap H_n = H} m_1\left(H_1\right) \cdot m_2\left(H_2\right) \cdots m_n\left(H_n\right) \end{aligned} \tag{24}$$

Where, the parameter K can be expressed as:

$$K = \sum_{H_1 \cap H_2 \cap \cdots \cap H_n \neq H} m_1(H_1) \cdot m_2(H_2) \cdots m_n(H_n) \tag{25}$$

However, due to the large amount of fusion data in the multi-agent system, the calculation recursion of the combination of two evidences is used for equivalent substitution.

5.2 Global Trust Model

The overall trust assessment is calculated by assigning appropriate weights to the direct and indirect trust values derived from the preceding analyses. After the end of the interactive time slot t, the global trust update function of vehicle i to Subgrade processing unit RSU_j can be defined as:

$$Trust_{i,j}(t) = \varpi DTrust_{i,j}(t) + (1-\varpi) ITrust_{i,j}(t) \tag{26}$$

Where, parameter ϖ is the weight coefficient

To sum up, the calculation task generated by vehicle i will be completed through the following four steps:

1) When the task unloading strategy of the vehicle is $\alpha_t^{i,0} = 1$, it means that the task of vehicle i in time slot t is processed and calculated locally; When the strategy is $\alpha_t^{i,j} = 1$ and $j \neq 0$, the on-board task will be assigned to the subgrade processing unit RSU_j for calculation.

2) The subgrade processing unit RSU_j processes the calculation task from each vehicle unloading according to the FIFO strategy.

3) After the on-board task calculation is completed, the corresponding calculation results will also be returned to the corresponding vehicle.

4) After the end of the current time slot t, the vehicle i updates its trust in all subgrade processing units. Where, if vehicle i interacts with subgrade processing unit RSU_j, i.e. $\alpha_t^{i,j} = 1$ and $j \neq 0$, the trust value is updated according to the global trust formula; If the vehicle task is completed locally, it means that the two have not communicated. The latest interaction experience is used to update the trust value of this time slot, i.e. $Trust_{i,j}(t) = Trust_{i,j}(t-1)$.

Since the objective for every vehicle is to reduce the time duration and energy expenditure associated with executing its specific processing workload, the optimization goal of this chapter is to reasonably unload the task and allocate the computing resources in the subgrade unit to make the whole model reach the overall optimization. Therefore, the optimization problem P of the trusted task unloading model proposed in this chapter can be described by equation (27).

$$\begin{aligned} P: &\max \sum_{i=1}^{n} \left(\alpha \frac{\overline{t} - T_i}{\overline{t}} + \beta \frac{\overline{e} - E_i}{\overline{e}} \right) \\ subject\ to: &C1: \sum_{j=0}^{m} a_t^{i,j} = 1, i \in \{1, 2, \cdots, n\} \\ &C2: T \leq \tau_{\max} \\ &C3: Res_{i,j} \leq Res_j \end{aligned} \tag{27}$$

Wherein, the parameters $\overline{t}$ and $\overline{e}$ respectively represent a basic value of the latency and energy usage of the calculation task. The two are introduced to unify the calculation units of the two dimensions of delay and energy. Parameters α and β represent the weight factors of delay and energy respectively, and the sum of the two is 1. When parameter α is 1, the formula indicates that the current system only considers the influence of time

delay. Considering that there are more and more new energy vehicles with relatively limited energy in our lives, we should try our best to save the energy used by vehicles to communicate and calculate the tasks generated by vehicles. Therefore, the size of parameter β depends on the remaining energy E_r of the vehicle during this time interval, so as to save more energy by sacrificing time efficiency on the basis of completing the task. The two maintain a correlation, $\beta \propto E_r$. at the same time, the value of parameter β set in this paper remains unchanged for a certain distance, and parameter E_r follows a truncated normal distribution. In addition, in the following discussion, the description of vehicle energy is to describe the parameter E_r .

Constraint C1 indicates that the task is indivisible and can only be completed by the vehicle itself or migrated to a subgrade processing unit; Constraint C2 indicates that the calculation needs to be completed within its maximum allowable delay when processing a task; The constraint C3 indicates that the amount of resources allocated during the calculation of the task unloaded to the subgrade unit cannot exceed the total amount of resources. In this chapter, an algorithm based on deep reinforcement learning is used to solve the NP hard combinatorial optimization problem.

6 TBMADDPG Algorithm Model

Because some subgrade units are maliciously connected to the vehicle Edge network, if the vehicle task is blindly unloaded to the malicious edge node, it may lead to problems such as the task feedback is not timely or the task does not respond, and then reduce the user experience of vehicle users. Because the system needs to calculate and make unloading decisions for all vehicle tasks generated in each time slot of the model, and the computing resources provided by the subgrade processing unit are dynamic in different time slots, these will lead to changes in the unloading decisions of the model system. Effectively identifying malicious subgrade units while simultaneously balancing edge resource allocation for task computation and offloading poses a significant challenge. To address this, the present chapter introduces TBMADDPG, a distributed algorithm for resource scheduling. This novel approach integrates a trust evaluation framework with deep reinforcement learning techniques. A key innovation lies in incorporating a dynamic trust coefficient within the resource allocation model. This mechanism enables the system to rapidly and reliably detect malicious infrastructure elements and concurrently decrease overall time and energy expenditure.

At the outset, baseline trust for all subgrade units is set by each vehicle. Following this initialization, trust values are updated per time slot based on the previously described analysis (Section X). The resulting global trust across all vehicles and subgrade units is represented by the matrix:

$$TrustMatrix(t) = \begin{bmatrix} Trust_{1,1}(t), Trust_{1,2}(t), \cdots, Trust_{1,m}(t) \\ Trust_{2,1}(t), Trust_{2,2}(t), \cdots, Trust_{2,m}(t) \\ \vdots \quad , \quad \vdots \quad , \cdots, \quad \vdots \\ Trust_{n,1}(t), Trust_{n,2}(t), \cdots, Trust_{n,m}(t) \end{bmatrix} \tag{28}$$

For any subgrade unit within a specific time interval, its trust coefficient represents the collective average of global trust scores received from all vehicles. Mathematically, this can be stated as:

$$Trust_j(t) = \sum_{i=1}^{n} Trust_{i,j}(t) \Big/ n \tag{29}$$

6.1 Vehicle Networking System State Space

In the model, the vehicle will randomly generate different types of tasks at the beginning of each time slot. At the same time, the computing resources provided by the subgrade unit to the vehicle are also dynamic, and its allocation rate is recorded as T_R^t. Additionally, the global trust matrix $T\text{rust}M\text{atrix}$（t）undergoes adjustments

during every time step, reflecting the exchanges occurring between vehicles and subgrade units. Therefore, the system state space of the task unloading model based on trust model needs to consider the task information, waiting delay information and global trust information. Depending on the origin of the data, distinguishing between information generated by the vehicle itself and system-wide global data, first define the status information of the vehicle itself as:

$$s_i(t) = \left[\tau_{\max}, E_{r,i}, t_{w,i} \right] \tag{30}$$

For the present time slot *t*, the state representation of the system comprises:

$$S_t = \left(s_1(t), s_2(t), \cdots, s_n(t), t_{w,m}, T_R^t, TrustMatix \right) \tag{31}$$

Where parameter $E_{r,i}$ represents the energy of vehicle *i*; $t_{w,i}$ is the queuing delay of vehicle *i*; The parameter $t_{w,m}$ represents the 1×*m*-dimensional queue delay matrix of all subgrade units.

6.2 Action Space of Internet of Vehicles System

After acquiring the state space information of the environment, the system selects the next action to execute based on the policy defined within the policy network, thereby determining the subsequent state space. Considering that in the model, tasks for all vehicles are either completed by the vehicle itself or are offloaded to a subordinate computational unit for processing, the unloading decision for a specific vehicle j in the current time slot *t* can be represented by a matrix with dimensions $(m+1)\times 1$:

$$A_i^t = \left[a_{i,0}^t, a_{i,1}^t, a_{i,2}^t, \cdots, a_{i,m}^t \right]^T \tag{32}$$

Since vehicle tasks are indivisible, each task must uniquely correspond to a single action; specifically, formula (32) must satisfy the subsequent criteria:

$$\sum_{j=0}^{m} a_{i,j}^t = 1, i \in \{1, 2, \cdots, n\} \tag{33}$$

To sum up, the dynamic space of the system can finally be expressed as an $(m+1)\times n$ dimensional matrix.

According to formula (33), it can be ascertained that within a single time slice, the entire model contains $(m+1)^n$ selectable actions. As the number of vehicles and subordinate computational units in the model increases, the action space demonstrates exponential growth, leading to an excessively large space scale and subsequently triggering the curse of dimensionality problem. To address this challenge, this paper employs a multi-agent algorithm to seek a solution.

$$\begin{aligned} A_t &= \left[A_1^t, A_2^t, \cdots, A_n^t \right] \\ &= \begin{bmatrix} a_{1,0}^t, a_{2,0}^t, a_{3,0}^t, \cdots, a_{n,0}^t \\ a_{1,1}^t, a_{2,1}^t, a_{3,1}^t, \cdots, a_{n,1}^t \\ a_{1,2}^t, a_{2,2}^t, a_{3,2}^t, \cdots, a_{n,2}^t \\ \vdots, \vdots, \vdots, \cdots, \vdots \\ a_{1,m}^t, a_{2,m}^t, a_{3,m}^t, \cdots, a_{m,2}^t \end{bmatrix} \end{aligned} \tag{34}$$

6.3 System Reward Function

The principal aim of the system is managing the offloading and computation of vehicle tasks while adhering to specified constraints. It simultaneously seeks to minimize the time required for completing these tasks and to preserve the stability associated with the virtual energy queue. Regardless of the action a vehicle executes, the system furnishes feedback consistent with its configured parameters. When an effective action leads to the successful accomplishment of a task, a reward is subsequently granted, which is characterized by the following formulation:

$$r_k(t) = \alpha \frac{\overline{t} - T_k}{\overline{t}} - \beta E_n(t) E_n \tag{35}$$

The specific weighting coefficient utilized within the reward function adopts its value directly from formula (27). However, should the agent select an incorrect action, thereby preventing the task from being successfully computed and its results returned, the system incurs a penalty. This penalty amount is determined as specified by the definition below:

$$PF = \{pf_1, pf_2, pf_3\} \tag{36}$$

These three specified penalty amounts correspond to failures associated with distinct categories of tasks. Moreover, as the assessed significance of a particular task diminishes, the magnitude of the penalty incurred for its failure is correspondingly reduced.

Consequently, the formulation for the reward function utilized in this chapter is specified as follows:

$$r_t = \max\left(\sum_{k=1}^{K} \chi_k r_k(t) + \sum_{k=1}^{K} (1 - \chi_k) PF_k\right) \tag{37}$$

Where, χ_k=1 indicates that task *k* has been successfully processed; χ_k=0 indicates that the system selected the wrong policy when processing task *k*.

Consistent with the problem description P in formula (27), this chapter's primary optimization aim involves minimizing the system's time and energy expenditure for completing vehicle tasks. However, because this work addresses task unloading and resource allocation strategies in settings where trust levels can vary, it is necessary to account for the credibility of 'subgrade units'. Consequently, trust values are integrated as an essential incentive within the reward function, which this chapter ultimately defines as follows:

$$R_t = \max\left(\sum_{i=1}^{n} (1 + \xi_{i,j}(t) Trust_j(t)) \left(\alpha \frac{\overline{t} - T_i}{\overline{t}} + \beta \frac{\overline{e} - E_i}{\overline{e}}\right) + \sum PF\right) \tag{38}$$

Where, parameter $\xi_{i,j}(t)$ is a discrete variable, this signifies the varying trust weighting factors produced by the system during the implementation of diverse strategies. When $a_{i,0}^t = 1$, it means that the task is completed locally in the vehicle. At this time, it is irrelevant to the trust model, and the parameter $\xi_{i,j}(t)$ is 0; When the subgrade unit selected by the strategy action $a_{i,j}^t$ is malicious, the size of parameter $\xi_{i,j}(t)$ is 1, and the reward feedback is reduced by increasing the consumption caused by the wrong strategy; When the normal working subgrade unit is selected, the size of parameter $\xi_{i,j}(t)$ is -1, that is, the reward feedback obtained at this time is large. To sum up, the trust weighting coefficient can be expressed by formula (39).

$$\xi_{i,j}(t) = \begin{cases} 0 & a_{i,0}^t = 1 \\ 1 & a_{i,j}^t = 1 \; and \; RSU_j \; malicious \\ -1 & a_{i,j}^t = 1 \; and \; RSU_j \; \text{normal} \end{cases} \tag{39}$$

This reward function formulation holistically integrates considerations of system latency, overall energy expenditure, and node trustworthiness levels. It specifically leverages trust scores as an incentive coefficient. This mechanism facilitates rapid identification of potentially malicious 'subgrade units' operating within the system, thereby aiming to accelerate the training convergence and enhance overall system performance.

6.4 Multi Agent Reinforcement Learning Algorithm

The direct application of single-agent algorithms to multi-agent scenarios faces the challenge of non-stationarity. Agents view peers as environmental components and store only their own experiences; thus, changes in others' strategies are perceived as unpredictable environmental shifts, leading to poor guidance and learning instability. Therefore, practical multi-agent approaches often assume agents (like Agent A) can observe peer actions or strategies, utilizing this external data for training. Maximum likelihood estimation serves as a method to estimate these peer strategies from observations. Furthermore, to handle potential overfitting due to evolving strategies, agents can adopt robust techniques like training a portfolio of diverse policies and subsequently blending or selecting from them to establish a more general strategy.

Set k agents in the process of centralized training, and the network parameters are $\theta=\{\theta_1,\theta_2,\cdots,\theta_K\}$ respectively; The deterministic strategies of all agents can be represented by set $\mu=\{\mu_{\theta_1},\mu_{\theta_2},\cdots,\mu_{\theta_K}\}$. So for the deterministic strategy μ_k of agent K, the gradient can be written as:

$$\nabla_{\theta_k} J(\mu_k)=\mathbb{E}_{S,A\sim D}\left[\nabla_{\theta_k}\mu_k(a_k\mid s_k)\nabla_{a_k}Q_k^{\mu}(S,a_1,a_2,\cdots,a_K)\mid_{a_k=\mu_k(s_k)}\right] \tag{40}$$

Where, the parameter D represents the experience playback pool, indicating that the experience tuples (S,A,S',R) participating in the training are all taken from the experience pool.

For Cirtic network, it can be updated according to the loss function:

$$Loss(\theta_k)=\mathbb{E}_{S,A,S',R}\left[\left(Q_k^{\mu}(S,a_1,a_2,\cdots,a_K)-y\right)^2\right] \tag{41}$$

$$y=r_k+\gamma Q_k^{\mu'}(S',a_1',a_2',\cdots,a_K')\mid_{a_j'=\mu_j'(s_j)} \tag{42}$$

The actor network's parameters can be updated by minimizing the agent's policy gradient:

$$\nabla_{\theta_k} J\approx\frac{1}{Z}\sum_j\nabla_{\theta_k}\mu_k(s_k^j)\nabla_{a_k}Q_k^{\mu}(S^j,a_1^j,a_2^j,\cdots,a_K^j)\mid_{a_k=\mu_k(s_k^j)} \tag{43}$$

The parameter j in the above formula represents the index of the extracted experience group.

To enhance the training process stability, the DDPG algorithm utilizes a "soft update" mechanism for adjusting specific parameters of its target network. This update rule is defined as follows:

$$\theta^{Q'}\leftarrow\tau\theta^{Q}+(1-\tau)\theta^{Q'} \tag{44}$$

This chapter proposes a distributed task unloading and resource scheduling algorithm, termed TBMADDPG, is founded upon a trusted model and deep reinforcement learning methodologies. The algorithm schematic diagram is shown in Fig. 5. After all agents execute the policy action, they will go through the trust evaluation model and make a global update after the end of the current time slot.

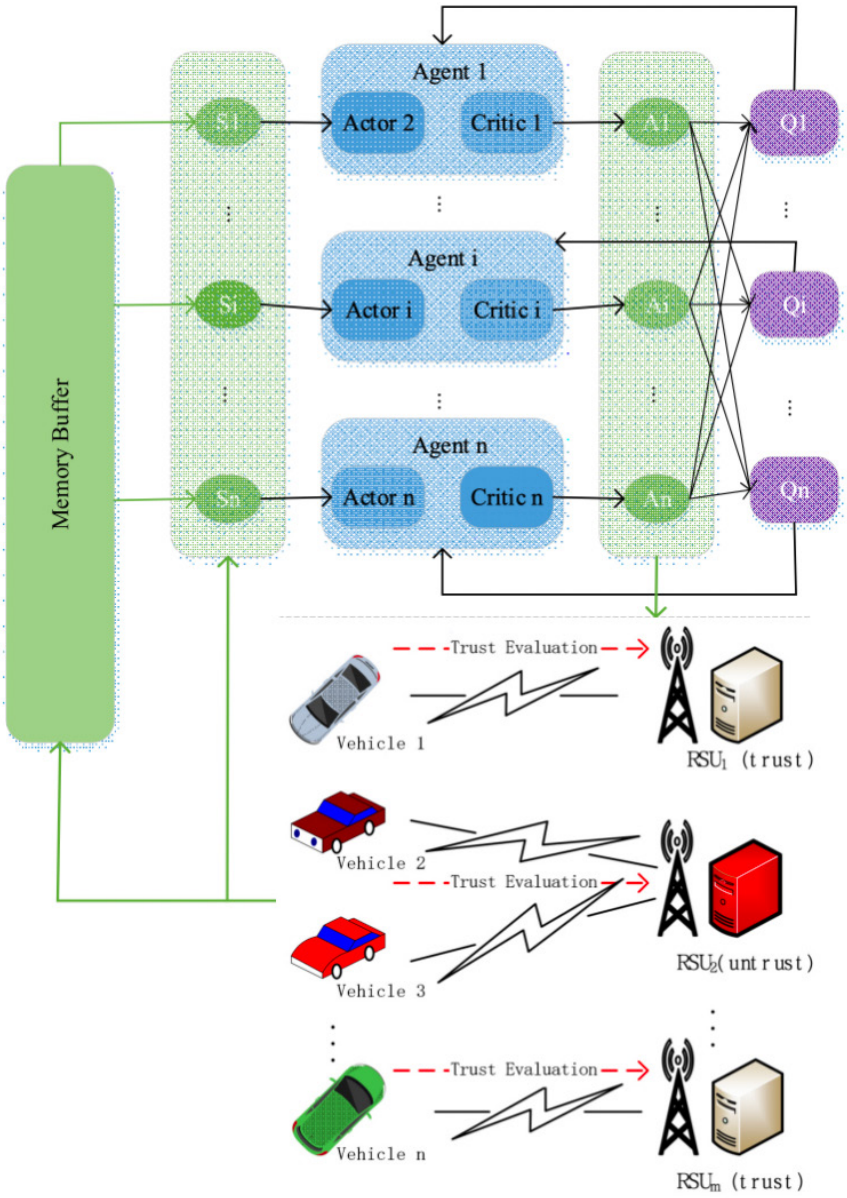

Fig. 5. The algorithm diagram of TBMADDPG

Table 1 illustrates the detailed training procedure for the TBMADDPG algorithm:

Table 1. TBMADDPG training algorithm description for N agents

TBMADDPG algorithm for n agents	
1	Randomly initialize all Actor and Critical networks and their respective weight parameters and their corresponding experience pools;
2	Initialize the information such as computing resources and network status in the system model;
3	Initialize the direct and global trust matrix, and randomly specify the index of malicious subgrade units;
4	**For** episode = 1 to max_ episodes:

Environmental parameters of the initial action exploration process: noise variable N_t, vehicle and subgrade unit information;
For time = 1 to max_ slots:
Randomly generate and sort the vehicle task information under the current time slot;
Update the resource allocation of subgrade units in the model;
For task = 1 to max_ indexes:
Each agent outputs actions according to the current strategy network, noise disturbance and restriction conditions: $a_i = \mu_{\theta_i} + N_t$;
Execute action $a=(a_1, a_2, \cdots, a_n)$ to get reward r and the next state s';
Update the waiting delay queue information and store the sample data (s, a, r, s') into the experience pool R;
End For
Calculate and update the global trust of all vehicles to the subgrade unit in the current time slot;
For agent = 1 to N:
Randomly sample Z pieces of sample data (s^j, a^j, r^j, s'^j) in the experience pool R to form a mini batch;
Update the online network parameters of Critical through equation (6-14);
Update the online network parameters of Actor through equation (6-16);
End For
Update the target network parameters of each agent through formula (6-17);
End For
End For

The performance of the trained system model can be tested by the test algorithm. The test algorithm is shown in Table 2:

Table 2. TBMADDPG testing algorithm description

TBMADDPG test algorithm
Load the network parameter θ_n, $n = \{1, 2, ..., N\}$ that the agent has trained;
Initialize the trust matrix and system state of the model;
For agent=1 to N:
Select action $A_t^n = \text{argmax}\, Q(s, a, \theta_n)$;
End For
Get the total revenue generated by all agents executing the task unloading strategy;

7 Conclusion and Prospect

This paper proposes a reinforcement learning algorithm, TBMADDPG, for identifying malicious road base units in urban intersection scenarios. The algorithm is based on the trust model and multi-intelligentsia, and it is implemented by centralized training and distributed execution. A comparative analysis is conducted with various algorithms to assess its efficacy. The TBMADDPG algorithm demonstrates a notable capacity to identify dynamic malicious behaviors in telematics, facilitating the development of enhanced task offloading and resource scheduling strategies. These strategies contribute to a substantial reduction in total time and energy consumption within the system. Additionally, the algorithm exhibits adaptive capabilities, enhancing its performance in comparison to traditional policy algorithms.

The following is a summary and outlook of some existing problems and future work according to the research of this paper:

1) For the simulations detailed in this study, the proposed algorithms were built upon the DDPG framework. Recognizing the inherent limitations of the DDPG algorithm, future research could explore the application of alternative reinforcement learning approaches, such as PPO and A3C, within the simulation environment. Such investigations might potentially lead to enhanced training efficacy or a more streamlined implementation process.

2) In this paper, the agent's behaviors are a direct consequence of the interaction between the vehicle and the subgrade unit. Real-world scenarios often involve a "re-unloading" condition, where the progression of specific actions necessitates evaluation by an independent third party. This scenario poses a significant challenge for reinforcement learning in its ability to derive an optimal policy, largely due to the potential for self-serving actions from these third-party entities. To overcome this difficulty, we suggest exploring the integration of the models detailed in Chapters 3 and 4. Furthermore, beyond the reward provided by the income function, incorporating a trust metric as a form of altruistic motivation could be considered. Such an approach may prove beneficial for reinforcement learning in addressing practical issues akin to "re-unloading".

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant 62173026.

References

[1] Z. Ning, Y. Feng, M. Collotta, X. Kong, X. Wang, L. Guo, X. Hu, B. Hu, Deep Learning in Edge of Vehicles: Exploring Trirelationship for Data Transmission, IEEE Transactions on Industrial Informatics 15(10)(2019) 5737–5746. https://doi.org/10.1109/TII.2019.2929740

[2] G. Sun, G.O. Boateng, D. Ayepah-Mensah, G. Liu, J. Wei, Autonomous Resource Slicing for Virtualized Vehicular Networks With D2D Communications Based on Deep Reinforcement Learning, IEEE Systems Journal 14(4)(2020) 4694–4705. https://doi.org/10.1109/JSYST.2020.2982857

[3] Y. He, C. Liang, F.R. 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. https://doi.org/10.1109/TNSE.2018.2865183

[4] Y. Wei, F.R. Yu, M. Song, Z. Han, Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning, IEEE Internet of Things Journal 6(2)(2019) 2061–2073. https://doi.org/10.1109/JIOT.2018.2878435

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge Computing: Vision and Challenges, IEEE Internet of Things Journal 3(5)(2016) 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[6] J. Zhang, H. Guo, J. Liu, Y. Zhang, Task offloading in vehicular edge computing networks: A load-balancing solution, IEEE Transactions on Vehicular Technology 69(2)(2020) 2092–2104. https://doi.org/10.1109/TVT.2019.2959410

[7] J. Zhang, H. Guo, J. Liu, Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme, Mobile Networks and Applications 25(5)(2020) 1736–1745. https://doi.org/10.1007/s11036-020-01584-6

[8] X. Huang, L. He, W. Zhang, Vehicle speed aware computing task offloading and resource allocation based on multi-agent reinforcement learning in a vehicular edge computing network, in: Proc. 2020 IEEE International Conference on Edge Computing (EDGE), 2020. https://doi.org/10.1109/EDGE50951.2020.00008

[9] H. Cho, Y. Cui, J. Lee, Energy-efficient cooperative offloading for edge computing-enabled vehicular networks, IEEE Transactions on Wireless Communications 21(12)(2022) 10709-10723. https://doi.org/10.1109/TWC.2022.3186590
[10] Y. Liu, H. Yu, S. Xie, Y. Zhang, Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks, IEEE Transactions on Vehicular Technology 68(11)(2019) 11158–11168. https://doi.org/10.1109/TVT.2019.2935450
[11] X. Li, A computing offloading resource allocation scheme using deep reinforcement learning in mobile edge computing systems, Journal of Grid Computing 19(3)(2021) 1–12. https://doi.org/10.1007/s10723-021-09568-w
[12] X. Zhu, Y. Luo, A. Liu, N.N. Xiong, M. Dong, S. Zhang, A deep reinforcement learning-based resource management game in vehicular edge computing, IEEE Transactions on Intelligent Transportation Systems 23(3)(2022) 2422–2433. https://doi.org/10.1109/TITS.2021.3114295
[13] M. Khayyat, I.A. Elgendy, A. Muthanna, A.S. Alshahrani, S. Alharbi, A. Koucheryavy, Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks, IEEE Access 8(2020) 137052–137062. https://doi.org/10.1109/ACCESS.2020.3011705
[14] J. Shi, J. Du, J. Wang, J. Wang, J. Yuan, Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning, IEEE Transactions on Vehicular Technology 69(12)(2020) 16067–16081. https://doi.org/10.1109/TVT.2020.3041929
[15] S. Raza, M.A. Mirza, S. Ahmad, M. Asif, M.B. Rasheed, Y. Ghadi, A vehicle to vehicle relay-based task offloading scheme in vehicular communication networks, PeerJ Computer Science 7(2021) e486. https://doi.org/10.7717/peerj-cs.486
[16] J. Zhao, M. Kong, Q. Li, X. Sun, Contract-based computing resource management via deep reinforcement learning in vehicular fog computing, IEEE Access 8(2019) 3319–3329. https://doi.org/10.1109/ACCESS.2019.2963051
[17] A.M. Seid, G.O. Boateng, B. Mareri, G. Sun, W. Jiang, Multi-agent drl for task offloading and resource allocation in multi-uav enabled iot edge network, IEEE Transactions on Network and Service Management 18(4)(2021) 4531–4547. https://doi.org/10.1109/TNSM.2021.3096673
[18] M. Hua, Y. Wang, C. Li, Y. Huang, L. Yang, Energy-efficient optimization for uav-aided cellular offloading, IEEE Wireless Communications Letters 8(3)(2019) 769–772. https://doi.org/10.1109/LWC.2019.2891727
[19] S. Li, X. Hu, Y. Du, Deep reinforcement learning for computation offloading and resource allocation in unmanned-aerial-vehicle assisted edge computing, Sensors 21(19)(2021) 6499. https://doi.org/10.3390/s21196499
[20] 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. https://doi.org/10.1109/TNSM.2021.3099472
[21] C. Zheng, S. Liu, Y. Huang, L. Yang, Hybrid policy learning for energy-latency trade off in mec-assisted vr video service, IEEE Transactions on Vehicular Technology 70(9)(2021) 9006–9021. https://doi.org/10.1109/TVT.2021.3099129
[22] 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. https://doi.org/10.1109/TETC.2019.2902661
[23] X. Zhu, Y. Luo, A. Liu, M.Z.A. Bhuiyan, S. Zhang, Multiagent deep reinforcement learning for vehicular computation offloading in iot, IEEE Internet of Things Journal 8(12)(2021) 9763–9773. https://doi.org/10.1109/JIOT.2020.3040768
[24] W.L. Tan, W.C. Lau, O. Yue, T.H. Hui, Analytical models and performance evaluation of drive-thru internet systems, IEEE Journal on selected areas in Communications 29(1)(2011) 207–222. https://doi.org/10.1109/JSAC.2011.110120
[25] Y. Hui, Z. Su, T.H. 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. https://doi.org/10.1109/TITS.2018.2873096
[26] Z. Su, Y. Hui, T.H. Luan, Distributed task allocation to enable collaborative autonomous driving with network softwarization, IEEE Journal on Selected Areas in Communications 36(10)(2018) 2175–2189. https://doi.org/10.1109/JSAC.2018.2869948
[27] 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(2018) 63868–63879. https://doi.org/10.1109/ACCESS.2018.2872740