electronics

MDPI

Article

Dynamic Selection Slicing-Based Offloading Algorithm for In-Vehicle Tasks in Mobile Edge Computing

Li Han, Yanru Bin *, Shuaijie Zhu and Yanpei Liu

College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China; hanli@zzuli.edu.cn (L.H.); 332207050717@email.zzuli.edu.cn (S.Z.); liuyanpei@zzuli.edu.cn (Y.L.)

* Correspondence: 332107040599@email.zzuli.edu.cn; Tel.: +86-13298332949

Abstract: With the surge in tasks for in-vehicle terminals, the resulting network congestion and time delay cannot meet the service needs of users. Offloading algorithms are introduced to handle vehicular tasks, which will greatly improve the above problems. In this paper, the dependencies of vehicular tasks are represented as directed acyclic graphs, and network slices are integrated within the edge server. The Dynamic Selection Slicing-based Offloading Algorithm for in-vehicle tasks in MEC (DSSO) is proposed. First, a computational offloading model for vehicular tasks is established based on available resources, wireless channel state, and vehicle loading level. Second, the solution of the model is transformed into a Markov decision process, and the combination of the DQN algorithm and Dueling Network from deep reinforcement learning is used to select the appropriate slices and dynamically update the optimal offloading strategy for in-vehicle tasks in the effective interval. Finally, an experimental environment is set up to compare the DSSO algorithm with LOCAL, MINCO, and DJROM, the results show that the system energy consumption of DSSO algorithm resources is reduced by 10.31%, the time latency is decreased by 22.75%, and the ratio of dropped tasks is decreased by 28.71%.

Keywords: mobile edge computing; network slicing; in-vehicle task offloading; deep reinforcement learning

Citation: Han, L.; Bin, Y.; Zhu, S.; Liu, Y. Dynamic Selection Slicing-Based Offloading Algorithm for In-Vehicle Tasks in Mobile Edge Computing. *Electronics* **2023**, *12*, 2708. https://doi.org/10.3390/electronics12122708

Academic Editors: Claus Pahl and Antonio Brogi

Received: 26 April 2023
Revised: 12 June 2023
Accepted: 15 June 2023
Published: 16 June 2023

1. Introduction

Mobile edge computing (MEC) involves deploying server nodes near users to address excessive latency issues and data flow congestion [1]. By enabling the deployment of functions and applications at the wireless access network (RAN) edge layer closest to user devices, MEC provides computing resources and high-frequency bandwidth to avoid network congestion caused by massive data. At the same time, user devices offload processing tasks to the network edge nodes and real-time access data from MEC edge nodes to achieve information exchange purposes [2,3]. However, the distributed deployment of MEC edge nodes and task offloading between nodes consume additional energy. Therefore, reducing the energy consumption of task offloading in the MEC environment while meeting the requirements of latency and quality of service (QoS) has become a significant challenge hindering the development of MEC applications.

The Internet of Vehicles (IoV) is one of the main application scenarios for task offloading in MEC [4]. In actual IoV applications, task offloading between in-vehicle terminal devices and edge servers is typically influenced by multiple factors [5,6]. Among them, considering only offloading tasks to edge nodes when confronted with a significant volume of service requests and a vast amount of data, it is inevitable that an imbalance in the load of edge nodes and data loss may occur. Furthermore, extensive data exchange can result in high time latency and energy consumption during communication, which will lead to mission failure. Especially in vehicular networks with heterogeneous latency and computing requirements because vehicles usually choose the closest or best communication

quality Base Station (BS) or Road Side Unit (RSU) to perform computing offloading of vehicular tasks, and the high mobility of vehicles and short effective communication range can lead to frequent switching of edge servers, which can easily cause untimely offloading and communication failures [7]. Thus, the offloading algorithm of vehicular tasks needs a dramatic reduction in delay and energy consumption.

Network slicing [8] is introduced into the IoV to provide an economical and efficient solution for vehicle. Network slicing is one of the key technologies of software-defined networking (SDN) [9], which is the logical division of the common physical infrastructure into multiple network slices through various network access technologies to provide guarantees for heterogeneous quality-of-service performance and maximize network operator optimization goals. Each slice can act as an independent end-to-end network to support the service requirements of various in-vehicle applications [10,11]. However, the resource-constraint result in infrastructure providers not being able to satisfy all service requests [12]. Therefore, to tackle the problem that resource-constrained vehicular terminal devices can hardly handle massive computational demands in real-time, the issue of data transmission failure due to frequent changes in the Vehicular Ad-hoc Networks (VANETs) environment [13]. It is an urgent choice to build a lower cost computational offloading model using a user-oriented network slicing approach to handle the computational offloading of tasks to achieve the optimization goal of low latency and low energy consumption.

Most of the current offloading algorithms focus on leveraging Network Function Virtualization (NFV) and Software Defined Network (SDN) technologies to minimize the delay and energy consumption associated with task offloading [14,15]. However, these algorithms seldom consider the dependencies between tasks dependencies. The high-speed movement of vehicles requires frequent exchange of resource state information, which triggers untimely information updates and leads to low offloading efficiency of in-vehicle tasks [16–19]. For this reason, this paper considers the dependency relationships between in-vehicle tasks and integrates network slices into the edge server (ES), uses Dueling Network for model training, and dynamically updates the selection slice results. The main contributions are as follows.

1. The Dynamic selection slicing-based offloading algorithm for vehicular tasks: To accomplish the optimization objective of decreasing the aggregate expense related to task offloading, the dynamic selection slicing-based offloading algorithm for in-vehicle tasks in MEC is proposed in this paper. The algorithm uses the Dueling Network combined with Deep Q-Network (DQN) in deep reinforcement learning to dynamically select the optimal slicing results and then update the optimal offloading policy in the effective interval.
2. Performance Evaluation: The results show that the DSSO algorithm proposed in this paper improves the efficiency of multi-dimensional resource utilization and task completion rate and reduces the total cost of the offloading task compared with the LOCAL, MINCO, and DJROM.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 designs the in-vehicle tasks offloading model based on the dynamic selection slices. The DSSO algorithm is proposed in Section 4. Experimental comparison analysis is performed in Section 5. Finally, Section 6 is the conclusion of the paper.

2. Related Work

The rise of MEC facilitates mobile devices to offload tasks to nearby edge nodes for processing to reduce task time delays and decrease the discard ratio of delay-sensitive tasks [20–25]. Bi et al. [20] considered wireless power supply weighting and computational rate maximization in the MEC scenario and proposed a joint optimal offloading algorithm based on the alternating direction multiplicative decomposition technique, but the algorithm ignored the inter-task waiting delay during task offloading. Tang et al. [21] explored a distributed offloading algorithm for the Mobile Edge Computing (MEC) environment that relies on model-free deep reinforcement learning. The aim of this algorithm is to minimize

the expected long-term cost. Cheng et al. [22] focused on reducing the UE processing delay and energy consumption for offloading tasks and proposed a distributed offloading dynamic optimization algorithm for the MEC server multi-user, but the algorithm did not consider the optimization of the algorithm using radio resources and extend to other task-offloading scenarios with the complex structure of multi-user and multi-cell. Kim et al. [23] proposed a predictive model to decompose and offload tasks to multiple MEC servers to reduce the execution time at each MEC server. However, they did not consider the complexity of real WANs and the queueing delay of MEC servers when tasks are queued. Li et al. [24] designed an online energy minimization algorithm for multiple IoT devices with latency constraints in edge computing environments. Poularakis et al. [25] studied the multi-dimensional limitations of storage, computation, and communication to support MEC service placement and joint optimization problems for request routing; however, the algorithm does not consider the coordination problem of the existence of backhaul links between the base stations.

Network slicing is applied to computational offloading. The task improves the utilization of multi-dimensional resources in the network by selecting the best performance slices for computational offloading [26–28]. Chen et al. [26] proposed a DDQN-based computational offloading strategy for MEC environments that considers the time-varying channel quality of users and base stations, the energy units received by the wireless environment; thus, the maximized long-term utility of the dynamically exploited network slices resulting from the arrival of computational tasks. Huynh et al. [27] studied a network slicing-based management framework that considers the uncertainty and timeliness of network resource requests and captures the sliced requests through a semi-Markov decision process to obtain the optimal resource allocation policy. Al-Khatib et al. [28] proposed a slicing scheme that is based on priority and reservation. The scheme is designed for diverse vehicle applications and enhances the utilization of network resources while guaranteeing network utility. However, this approach does not account for the distinction between prediction models and immediate allocation.

The offloading algorithm based on deep reinforcement learning (DRL) provides a prospective solution for addressing tasks offloading in vehicular networks [29–33]. Peng et al. [29] proposed a distributed DRL-based algorithm for network resource management in highly dynamic vehicle scenarios. The algorithm trains deep neural networks with multiple agents, where the MEC server acts as the agent, to make online decisions on vehicle computation offloading and resource allocation. Mlika et al. [30] introduced a model-free approach based on DRL to solve the problem of the wireless channel, power allocation, slice selection, and vehicle grouping in MEC-enabled IoV networks using non-orthogonal multiple access (NOMA) to make better use of scarce channel resources. Nassar et al. [31] designed a DRL-based computational offloading strategy that allocates limited resources to vehicles with heterogeneous delay requirements. Li et al. [32] presented a task partitioning and scheduling algorithm to solve workload allocation, but relying on task partitioning and scheduling alone cannot find the optimal offloading strategy in an effective interval. Huang et al. [33] proposed an online offloading algorithm that maximizes the weighted value and computational efficiency based on a DRL-based algorithm, but the mobility of the device can make the algorithm harder to converge.

Although the above algorithms consider offloading tasks to edge nodes in MEC to improve task offloading efficiency as much as possible, the requirements of relational-dependent and delay-sensitive tasks can hardly be satisfied. The high-speed movement of vehicles requires frequent exchange of resource status information, which leads to untimely information updates and low offloading efficiency of in-vehicle tasks. Therefore, the offloading algorithm based on dynamic selection slicing for vehicle tasks is proposed where the custom slice is applied to the algorithm.

3. System Model and Optimization Goal

3.1. Communication Model

The process of in-vehicle task offloading based on the dynamic selection slices in MEC is shown in Figure 1. First, the in-vehicle terminal device initiates a compute offload request to the SDN controller. The SDN controller and the slice orchestrator interact with information by providing available status through the RSU, which generates data results that are the criteria for determining whether to offload the pending tasks to the edge server. Then, the offloading task is offloaded to a suitable slice within the edge server. Finally, the offloading results are transmitted to the slices through the offloading decision, and the optimal policy for task offloading is returned to the in-vehicle terminal device. The parameters involved in this paper are listed in Table 1.

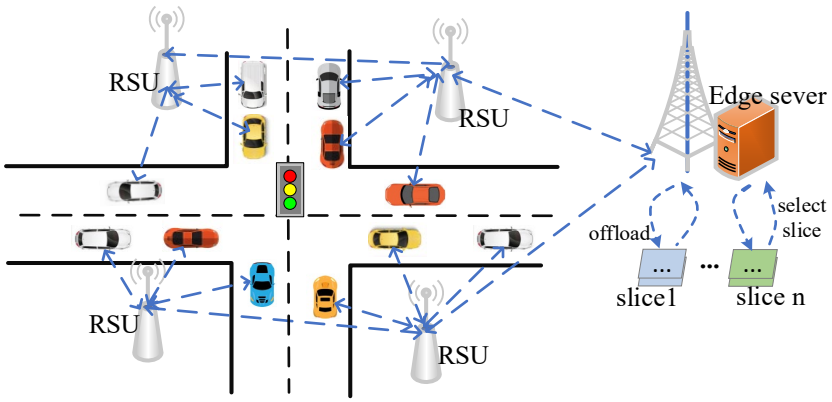

Figure 1. The process of in-vehicle task offloading.

Table 1. Parameters and Description.

Parameters	Description
D_i	Vehicle i
ω_i	In-vehicle task
d_{ω_i}	In-vehicle task input size (MB)
S_l	Slice l
AP_j	Access point j
y_i^l	An indicator variable to represent whether the vehicle D_i select slice S_l offload or not
T_{i,ω_i}^{ex}	The execution delay in the slice for vehicle D_i (S)
$T_{i,j,\omega_i}^{ex,l}$	The task execution time delay (S)
$T_{\omega_1,\omega_i}^{delay}$	The communication delay (S)
T_i^l	The total time latency (S)
E_i^{ex}	The execution energy in the slice for the vehicle D_i (J)
$E_i^{service}$	The energy consumption for the tasks ω_i to the slice S_l (J)
$E_i^{trans,l}$	The energy consumption for the vehicle D_i to transfer the task to the slice S_l (J)
E_i^l	the total energy consumption (J)
f_i^{local}	The CPU cycle frequency of the vehicle D_i (MHz)
P_i	The vehicle D_i transmission power (W)
$\tau(t)$	Task deadline (S)
σ	The noise in the channel transmission (dB)
Com_l	The total amount of computing resources in the slice S_l
λ_i^l	The number of computing resources allocated to the vehicle D_i in the slice S_l
$R_{i,j}^l$	The transmission rate (bps)
$B_{i,j}$	The network bandwidth (bit/s)
$h_{i,j}$	The channel gain between the vehicle D_i and the access point AP_j

3.2. Offloading Model

The Dynamic Selection Slicing-based Offloading Algorithm for in-vehicle tasks, $\omega = (\omega_1, \ldots \omega_{i-1}, \omega_i)$ represents vehicular tasks, and $x_i \in [0,1]$ indicates the offload strategy of the in-vehicle task. The dependencies of the in-vehicle tasks are described using the directed acyclic graph 2, and the nodes in Figure 2 show the computational offload tasks generated by the in-vehicle terminal devices, and the dependencies are represented between tasks using the directed edges of nodes. The node ω_1 without a parent is considered to be the start of a task, while the node ω_5 without a child is considered to be the end of a task, and the node ω_4 is represented as dependent on ω_2 and ω_3. The white nodes state that the edge server is capable of offloading tasks; the shaded nodes indicate that the task can only be executed locally in the vehicle.

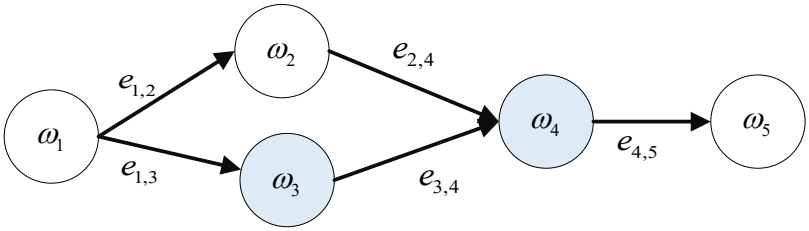

Figure 2. Task dependencies.

1. Local offload

$x_i = 0$ indicates that the task can only be executed locally in the vehicle. C_{ω_i} expresses various resources required per bit of a task, and M shows the set of vehicles. Therefore, the execution time of in-vehicle tasks T_i^{ex} can be expressed as:

$$T_i^{ex} = d_{\omega_i} \cdot C_{\omega_i} / f_i^{local} \quad \forall i \in M \tag{1}$$

when local execution is selected for vehicular tasks, the transmission time latency can be ignored. Total time latency T^{local} is the execution time, which can be shown as:

$$T^{local} = T_i^{ex} \quad \forall i \in M \tag{2}$$

The energy consumption E^{local} of the vehicle D_i processing task is represented as:

$$E^{local} = E_i^{ex} = d_{\omega_i} \cdot C_{\omega_i} \cdot \left(f_i^{local}\right)^2 \quad \forall i \in M \tag{3}$$

According to Equations (1)–(3), it can be seen that the task latency and energy consumption are mainly determined by the CPU cycle frequency.

2. Offloading to the slice

$x_i = 1$ represents that vehicle load pending tasks to the slices to access edge layer services. The vehicle terminal is connected to the slices S_l through several access points AP_j. Owing to the dependencies between most of the tasks, it is commonly the situation that AP_j is too busy to accept the next task. Thus, a binary variable $y_i^l \in (0,1)$ is introduced to determine whether the edge server interface is idle or not. When it is $y_i^l = 0$, the access point AP_j is idle. Otherwise, it needs to wait in queue.

In Equation (4), $\rho_{i,j}^l$ denotes the amount of wireless resources provided to the vehicle D_i by slice S_l docked to the access point AP_j, $B_{i,j}$ indicates the network bandwidth, P_i illustrates the transmission power of D_i, and $h_{i,j}$ indicates the channel gain between the vehicle D_i and the access point AP_j. "m" denotes a globally fixed constant, and similarly, P_m denotes the transmission power of D_m, $h_{m,j}$ shows the channel gain between D_m and

AP_j, and σ represents noise in the channel transmission. The transmission rate $R_{i,j}^{l}$ can be expressed as:

$$R_{i,j}^{l} = \rho_{i,j}^{l} \cdot B_{j,l} \cdot \log\left(1 + \frac{p_i \cdot h_{i,j}}{\sigma + \sum\limits_{y_{i,j}^{l}=1} p_m + h_{m,j}}\right) \tag{4}$$

The task transmission delay $T_{i,j,\omega_i}^{trans,l}$ is denoted as:

$$T_{i,j,\omega_i}^{trans,l} = T_{i,1,\omega_i}^{trans,j} = d_{\omega_i} / R_{i,1}^{l} \quad \forall R_{i,1,\omega_i}^{l} > 0 \tag{5}$$

Furthermore, the task execution time delay $T_{i,j,\omega_i}^{ex,l}$ is expressed as:

$$T_{i,j,\omega_i}^{ex,l} = T_{i,1,\omega_i}^{ex,l} = (d_{\omega_i} \cdot C_{\omega_i}) \Big/ \left(\lambda_i^{l} \cdot Com_l\right) \quad \forall \lambda_i^{l} > 0 \tag{6}$$

If the pending task is a dependent task when sliced for computational offloading inevitably results in a queueing situation. ω_{i-1} outputs the offloading result and returns to the slice, and then, ω_i is transmitted and offloaded to the slice, and δ_{ω_i} represents the offload result of the task ω_i. Therefore, it is necessary to consider the communication delays arising from the dependencies between tasks for latency-sensitive tasks. Through Formula (7), the communication delay $T_{\omega_1,\omega_i}^{delay}$ between tasks is expressed as:

$$T_{\omega_1,\omega_i}^{delay} = \begin{cases} \frac{\delta_{\omega_i}}{R_{i,j}^{l}} & x_i = 1, y_i = 0 \\ 0 & x_i = 1, y_i = 1 \end{cases} \tag{7}$$

The total time latency T_i^{l} for offloading the task ω_i to the slice S_l is shown as:

$$T_i^{l} = T_{i,j,\omega_i}^{ex,l} + T_{i,j,\omega_i}^{trans,l} + T_{\omega_1,\omega_i}^{delay} \tag{8}$$

The offloaded energy consumption $E_i^{service}$ for the tasks ω_i to the slice S_l is expressed as:

$$E_i^{service} = P_l \cdot \left(T_{i,j,\omega_i}^{ex,l} + T_{\omega_1,\omega_i}^{delay}\right) \tag{9}$$

Similarly, the energy consumption $E_i^{trans,l}$ required to support the vehicle D_i to transfer the task to the slice S_l can be given as:

$$E_i^{trans,l} = P \cdot T_{i,j,\omega_i}^{trans,l} \tag{10}$$

Therefore, the total energy consumption E_i^{l} for offloading tasks ω_i to slice S_l can be shown as:

$$E_i^{l} = E_i^{trans,l} + E_i^{service,l} \tag{11}$$

3. Optimization objectives

Throughout the overall Telematics system, energy consumption is a key metric to measure whether the offloading strategy is optimal. In this paper, the total vehicle energy consumption E_1 is expressed as:

$$E_1 = \sum_{i \in M} (1 - x_i) \cdot E_i^{ex} + \sum_{i \in M} \sum_{j \in O} \sum_{l \in P} x_i \cdot y_i \cdot E_i^{trans,l} \tag{12}$$

In Equation (12), $i \in M = \{1, 2 \cdots NM\}$, $j \in O = \{1, 2 \cdots NO\}$, $l \in P = \{1, 2 \cdots NP\}$, and N denotes the set of natural numbers.

Similarly, the energy consumption E_2 for offloading the in-vehicle tasks to the slice can be represented as:

$$E_2 = \sum_{i \in M} \sum_{l \in P} x_i \cdot y_i \cdot E_i^{ex,l} \tag{13}$$

Therefore, the total energy consumption E of the in-vehicle task offloading is shown as:

$$\text{P1}: \ E = E_1 + E_2 = \sum_{i \in M} (1 - x_i) \cdot E_i^{ex} + \sum_{i \in M} \sum_{j \in O} \sum_{l \in P} x_i \cdot y_i \cdot E_i^{trans,l} + \sum_{i \in M} \sum_{l \in P} x_i \cdot y_i \cdot E_i^{ex,l} \tag{14}$$

$$s.t. \begin{cases} C1: 0 \le f_i \le f_{i,\max}, 0 \le p_i \le p_{i,\max} \\ C2: 0 \le \sum\limits_{i \in M} \rho_{i,j}^l \le 1, \ \forall j \in O, l \in P \\ C3: x_i \le \sum\limits_{l \in P} y_i^l, x_i, y_i^l \in [0,1], \ \forall i \in M, j \in O, l \in P \\ C4: 0 \le \sum\limits_{i \in M} \lambda_i^l \le 1, \ \forall l \in P \\ C5: \lambda_i^l \le y_i^l \le \omega_i \lambda_i^l, \ \forall i \in M, l \in P \\ C6: (1 - x_i) \cdot E_i^{ex} + \sum\limits_{i \in M} \sum\limits_{j \in O} \sum\limits_{l \in P} x_i \cdot y_i^l \cdot E_i^l \le \tau_i, \ \forall i \in M \end{cases}$$

In Equation (14), constraints ($C1$)–($C3$) denote restrictions on the CPU cycle frequency, transmission power, and the amount of wireless resources. Constraint ($C4$) indicates the computational resources available to the slice. Constraint ($C5$) ensures that the task is processed before the deadline. Then, the NP-hard problem P1 is defined as Theorem 1.

Theorem 1. *P1 is an NP-hard problem.*

Proof of Theorem 1. Assuming that vehicles select a slice for offloading, the transmitted power is set to a maximum value. One access point AP, the deadline τ_i of the task is large enough, γ_1, γ_2 represent the average number of vehicles distributed equally, and allocate wireless resource $\rho_{i,j}^l$ and computing resources λ_i^l to vehicles, $\rho_{i,j}^l = 1/\gamma_1$, $\lambda_i^l = 1/\gamma_2$ in Equation (15), and the energy consumption $E_i^{l,*}$ can be expressed as:

$$E_i^{l,*} = \frac{\gamma_1 \cdot d_{\omega_i} \cdot p_{i,\max}}{B_{i,l} \cdot \log\left(1 + \frac{P_{i,\max} \cdot h_{i,j}}{\sum\limits_{m \in M} p_{m,\max} \cdot h_{m,j}}\right)} \tag{15}$$

In Equation (14), $x_i = 1$ represents the vehicle to offload the pending tasks to the slice. At this point, the entire system energy consumption can be expressed as shown in P2.

$$\text{P2}: \sum_{i \in M} \sum_{l \in P} y_i^l \cdot E_i^{l,*} \tag{16}$$

$$s.t. \begin{cases} C1: \sum\limits_{l \in P} y_i^l = 1, \ \forall i \in M \\ C2: \sum\limits_{i \in M} \sum\limits_{l \in P} y_i^l \cdot \rho_{i,1}^l \le 1 \end{cases}$$

In Equation (16), P2 is mapped to a multi-dimensional multi-choice knapsack problem (MMKP) [34], i.e., P2 is also an NP-hard problem. However, $x_i = 0$ indicates offloading of in-vehicle tasks to the vehicle terminal, i.e., the task tasks are processed locally. Thus, the system energy consumption can be expressed as $E^{local} = E_i^{ex}$. The above two cases can be reduced to P1: $E = E_1 + E_2 = \sum\limits_{i \in M} (1 - x_i) \cdot E_i^{ex} + \sum\limits_{i \in M} \sum\limits_{j \in O} \sum\limits_{l \in P} x_i \cdot y_i \cdot E_i^{trans,l} + \sum\limits_{i \in M} \sum\limits_{l \in P} x_i \cdot y_i \cdot E_i^{ex,l}$, and it can be concluded that P1 is an NP-hard problem. □

Similarly, the time latency T_1 of local vehicle D_i processing is expressed as:

$$T_1 = (1 - x_i) \sum_{i \in M} T_i^{ex} \tag{17}$$

The time delay for offloading the task to the slice T_2 can be shown as:

$$T_2 = \sum_{i \in M} \sum_{j \in O} \sum_{l \in P} x_i \cdot y_i^l \left[T_{i,j,\omega_i}^{trans,l} + T_{i,j,\omega_i}^{ex,l} + \left(1 - Z_{i,j}^l\right) T^{delay} \right] \tag{18}$$

In Equation (18), $Z_{i,j}^l \in [0,1], i \in M, j \in O, l \in P$, and $z_{i,j}^l$ indicates whether the vehicle D_i selects slice S_l and uses the access point AP_j to provide the amount of wireless resources. $z_{i,j}^l = 0$ indicates that the pending resources in the vehicle D_i have been successfully offloaded to the slice S_l; $z_{i,j}^l = 1$ represents that the access point is busy and needs to wait in the queue.

The total time delay of the system can be calculated using Equation (19).

$$\text{P3}: \ T = T_1 + T_2 \tag{19}$$

$$s.t. \begin{cases} C1: \rho_{i,j}^l \le Z_{i,j}^l \le \omega_i \rho_{i,j}^l, \ \forall i \in M \\ C2: y_i^l \le \sum\limits_{j \in o} Z_{i,j,\omega_i}^l, \ \forall i \in M, l \in P \\ C3: x_i, y_i^l, z_{i,j}^l \in [0,1], \ \forall i \in M, j \in O, l \in P \end{cases}$$

Since the transmission delay is 0 and the execution delay is relatively small when the task is selected for local offloading, the execution delay can be ignored in this model. Similarly, the P3 problem is reduced to P4.

$$\text{P4}: \ T = \sum_{i \in M} \sum_{j \in O} \sum_{l \in P} y_i^l \cdot T_{i,j,\omega_i}^{ex,l} + \left(1 - z_{i,j,\omega_i}^l\right) \cdot T^{delay} \tag{20}$$

$$s.t. \begin{cases} C1: \sum\limits_{l \in P} y_i^l = 1, \ \forall i \in M \\ C2: 0 \le z_{i,j,\omega_i}^l \le 1, \ \forall l \in P \\ C3: \sum\limits_{i \in M} \sum\limits_{j \in O} \sum\limits_{l \in P} T_{i,j,\omega_i}^{ex,l} + \left(1 - z_{i,l,\omega_i}^l\right) T^{delay} \le +\infty, \ \forall i \in M \end{cases}$$

In Equation (20), constraint ($C1$) guarantees that the slicing access point selected by the vehicle is free to receive the task ω_i; constraint ($C2$) ensures that the task ω_i successfully selects the access point AP_j for offloading the task; and constraint ($C3$) is to be sure that the task is processed before the deadline.

Therefore, the construction of the objective function F, the total cost of the task to offload the edge layer slices, and $F_{\min}$ can be formulated as:

$$F_{\min} = \beta_t E + \beta_e E \tag{21}$$

In Equation (21), β_t and β_e, respectively, denote the weighting factors of delay and energy consumption.

In a realistic in-vehicle task offload environment, the objective function F must be non-static. Therefore, the in-vehicle task offloading problem can be regarded as a mixed shaping nonlinear problem (MINLP). By the above proof, it can be seen that P2 and P4 are NP-hard problems whose feasible solutions are nonconvex, the time complexity grows exponentially with the growing number of tasks, and it is clear that the traditional methods cannot make adaptive decisions based on the dynamic characteristics of vehicles in the Telematics environment. Consequently, this paper proposes an in-vehicle task offloading algorithm based on the dynamic selection slices is proposed in this paper.

4. The Dynamic Selection Slice Offload Algorithm for In-Vehicle Task

4.1. Resource Slicing Based on Markovian Dynamic Adjustment

The offloading problem for the in-vehicle tasks oriented in MEC can be summarized as a sequential decision problem, and the Markov decision process (MDP) precisely provides an excellent mathematical model for sequential decision-making. In this paper, the optimization of in-vehicle task offloading is converted into finding the optimal solution in a Markov decision process and the design idea is as follows: First, the RSU in the connected vehicle environment provides available status $S(t)$, and the agent performs placement decision actions $A(t)$. Then, the agent obtains the reward value $r(t)$ based on the current state $S(t)$ and action $A(t)$, updates the neural network parameters by Dueling DQN, and takes the next action $A(t+1)$. Finally, the DRL algorithm maps the state space to the action space by iteratively selecting actions that maximize the expected future reward, using the Q-learning algorithm $f : S \rightarrow A$, generating the slicing results to maximize the reward value, i.e., minimizing the expected cost. Based on the Markovian dynamic adjustment of the state space, action space, and reward, the specific explanation is as follows:

1. State

The state space is defined as the collection of all possible states that the agent can be in $S(t) = \{H(t), d_{\omega_i}(t), \phi_{\omega_i}(t), \tau(t)\}$, and in this study, it includes the current status of network resources, such as wireless channel $H(t)$, input size $d_{\omega_i}(t)$, various required resources ϕ_{ω_i}, and deadline time $\tau(t)$, available CPU as well as the current status of each slice and the vehicular location.

2. Action

The action space is the set of actions taken in the available states of the MEC-oriented vehicle networking when the agent processes the vehicular task ω_i in a given time slot t corresponding to a certain state $S(t)$. The offloading decision action $A(t)$ was taken by the vehicle. To offload the vehicular task to the appropriate slice for execution, the action space is mapped to the set of slices within the edge server and denoted as $A_{\omega_i}(t) = \{1, \cdots, 0, 0\}$. Among them, $A_{\omega_i}(t) = \{1, \cdots, 0, 0\}$ represents offloading the in-vehicle task to the first slice for execution. Therefore, the action space for the in-vehicle task can be defined as: $A_{\omega_i}(t) = \{A_{\omega_1}(t), \cdots, A_{\omega_{i-1}}(t), A_{\omega_i}(t)\}$.

3. Reward

The reward provides feedback to the agent about the action taken in a given state, and its positive or negative value reflects whether there is an advantage in MEC when available state information $S(t)$ makes offloading decision actions $A(t)$ or not. If the reward value is negative, it indicates the need for dynamic adjustment, which needs to interact with the current in-vehicle network in time to obtain an enormous reward value. Meanwhile, the reward value can be used as an indicator to evaluate the merit of the offloading strategy. Additionally, the reward value $r(t)$ can serve as an indicator for evaluating the quality of offloading strategies.

4.2. Dueling DQN-Based Dynamic Selection Slice Offload Algorithm

In this section, based on the Markov model of resource slicing obtained in Section 4.1, the Dueling DQN in deep reinforcement learning is used to dynamically select the slicing results and then update the optimal offloading strategy. Among them, Dueling DQN is a combination of Dueling Network and DQN algorithm, and the architecture is shown in Figure 3. Firstly, the input of Dueling DQN is the feedback of state information by using deep convolutional neural networks to extract information such as shoulder obstacle image features and visual field features around the vehicle. Secondly, the DQN algorithm decomposes the action value function into the state value V and the advantage value A associated with each action in the state by introducing a dyadic network with two fully connected layer branches. Finally, the final Q-network output is obtained by combining these values linearly.

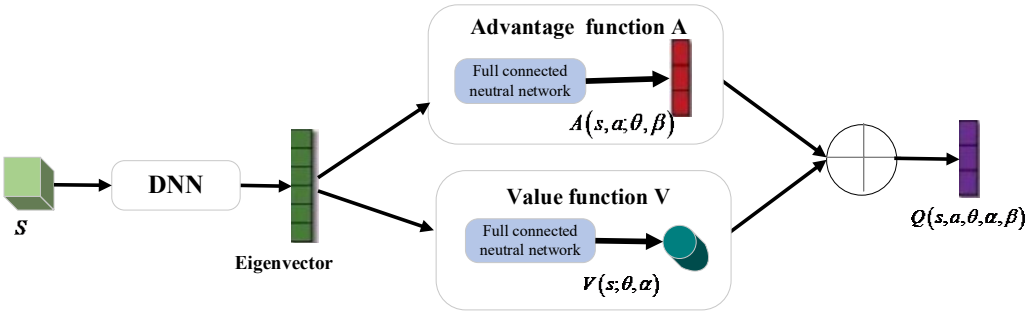

Figure 3. Dueling DQN architecture.

In Dueling DQN, after the agent executes the action a, the state is changed from s to s'. The reward value r is returned and stored in the experience pool. The sampling quaternions (s, a^*, r, s') from the experience pool for data training are based on the priority sampling method. Due to the dominant function $E_{a\sim\pi(s)}[A^{\pi}(s,a)] = 0$, when the selected action a has zero advantage, the stability of the DSSO algorithm is improved by using decentralized operations. Instead of using the maximum value as in the DQN algorithm, the average value is used in the proposed algorithm, and the forward mapping is implemented using the last module of the network to obtain the true value $Q(s,a,\theta,\alpha,\beta)$ denoted as:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\alpha) + A(s,a;\theta,\beta) - \frac{1}{|A|}\sum_{a*\in A} A(s,a^*;\theta,\beta) \tag{22}$$

In Equation (22), θ denotes the network parameters, α, β represents the relevant parameters of the fully connected layer network, a^* is the next action of a, and the corresponding state value of a^* is s'.

The DSSO algorithm predicts the total cost for in-vehicle task offloading, which can be expressed as:

$$y^{DuDQN} = r + \gamma \max_{a*} Q\left(s', a^*; \theta_i^-, \alpha^-, \beta^-\right) \tag{23}$$

The loss function L_i^{DuDQN} is defined as the discrepancy between the predicted value y^{DuDQN}, obtained from the output of the neural network, and the true value $Q(s,a,\theta,\alpha,\beta)$, which can be shown as:

$$L_i^{DuDQN} = \mathrm{E}_{\{s,a,s',r\}}\left[\left(y^{DuDQN} - Q(s,a;\theta,\alpha,\beta)\right)^2\right] \tag{24}$$

In Equation (23), θ_i^- denotes the update of the network parameters, and α^-, β^- states the update of the parameters related to the fully connected layer network. The parameter θ are continuously copied to the Target network through the Evaluation network under the optimization step C. The neural network is then subjected to backpropagation using the loss function L_i^{DuDQN} to update the parameters by comparing the magnitude of the reward value $r(t) = \varphi/(\beta_t T + \beta_e E), \varphi > 0$, when offloading a vehicle during a task to evaluate the effectiveness of offloading strategies.

4.3. Implementation of Offload Algorithm Based on the Dynamic Selection Slices

4.3.1. The Offload Algorithm Description

The DSSO algorithm pseudo code is shown in Algorithm 1.

Algorithm 1: The Dynamic Selection Slice Offloading Algorithm for In-Vehicle Task

Input: wireless channel status $H(t)$, input size $d_{\omega_i}(t)$, task deadline $\tau(t)$
In − Vehicle task ω_i and the maximum number of iterations ξ
Output: Task offload to slice strategy Ψ
1. Begin
2. Initialize the DNN parameters θ_1 and set the training interval δ
3. for $t = 1, 2, 3, \cdots, \xi$ do
4. if the slice access point is idle $y_i^l = 0$ then
5. Generate initial selection slice result $\{P_1(t), f_1(t), \rho_1(t), \eta_1(t)\}$
6. if $t \ mod \ \delta = 0$ then
7. Randomly select training sets $\{H(t), d(t), \tau(t)\}$;
8. Use Adam Optimizer to get the best training parameters θ;
9. end if
10. Get the computational resource solution in the valid interval $f_k(t)$ and $\eta_k(t)$;
11. obtain the radio resource allocation solution $\rho_k(t)$ and transmit power results $P_k(t)$ by dichotomous method;
12. Obtain selection slice result $\{P^*(t), f^*(t), \rho^*(t), \eta^*(t)\}$
13. else if the slice access point is busy $y_i^l = 1$ then
14. Wait and calculate the return to step 4
15. Calculating the reward value $r(t)$
16. Update Slice Offloading Policy Ψ
17. End

4.3.2. Algorithm Complexity Analysis

Algorithm 1 summarizes that lines 4–14 are based on offloading the pending tasks to the network slices of the edge server, i.e., $x_i = 1$, and the access point is free $y_i^l = 0$, and get selection slice results $\{P^*(t), f^*(t), \rho^*(t), \eta^*(t)\}$. Line 13 indicates that if the access point is busy, the waiting delay T^{delay} is recorded, and return to step 4. Judge whether this uninstallation solution is the best uninstallation strategy Ψ based on the reward value corresponding to the selection slice result.

In the DSSO algorithm, the time complexity is derived from the following components: (1) Time complexity of getting the selected slice section, which is $O(NM \cdot NP \cdot NO)$; (2) Time complexity of resulting resource allocation, which is $O(NM \cdot NP) + O(NM)$; (3) Time complexity of the resulting wireless resource allocation and transmit power setting, which is $O(NM \cdot NP \cdot NO)$. In summary, the overall time complexity is $O(NM \cdot NP \cdot NO)$.

5. Experimental Results and Analysis

5.1. Experimental Environment and Configuration

The performance of the proposed algorithm is evaluated by building an edge environment comprising an SDN controller, a slice orchestrator, and eight edge servers (ES). The end devices and servers were configured as listed in Table 2. The coverage area is 200×200 m^2, and equipment is evenly distributed throughout the coverage area, as depicted in Figure 4. Among them, the SDN controller installation is based on the Open Daylight platform [35], Kubernetes control the ES, the end-to-end orchestrator [36] is installed as a slicing orchestrator, and the test dataset for the experiments uses the KITTI dataset [37].

Table 2. Terminal device and server configurations.

Device Type	Memory	CPU
ES 1~2	4 GB	Intel(R) Core(TM) i5-10210U CPU @ 2.11 GHz
ES 3~4	4 GB	Intel(R) Core(TM) i7-2450M CPU @ 2.60 GHz
ES 5~6	4 GB	Intel(R) Core(TM) i5-4210M CPU @ 2.50 GHz
ES 7~8	4 GB	Intel(R) Core(TM) i5-4590U CPU @3.30 GHz
Dell Inspiron 5490	8 GB	Intel(R) Core(TM) i5-10210U CPU @ 1.60 GHz
Lenovo Xiaoxin pro 14	8 GB	Intel(R) Core(TM) i5-12500H CPU @ 1.80 GHz
Lenovo ThinkPad	8 GB	Intel(R) Core(TM) i7-5600U CPU @ 1.80 GHz
HP 840G3	16 GB	Intel(R) Core(TM) i5-8350U CPU @ 1.80 GHz
Xiaomi 11	8 GB	Snapdragon 778G
Honor 70 pro	12 GB	Dimensity 8000
Galaxy Z Flip3	16 GB	Snapdragon 888
Samsung w23	16 GB	Snapdragon 8 + Gen1

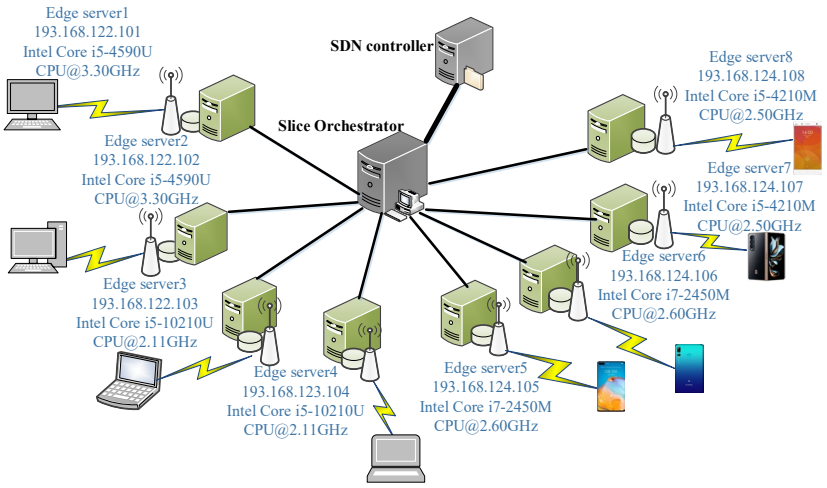

Figure 4. Experimental environment configuration.

The basic setup of the experiment is as follows: the vehicle detection task is assigned to two network slices with 28 vCPU cores and 20 MHz bandwidth, and the experiment adopts the method of controlling variables, which means that only one variable is changed in each experiment while keeping other variables the same. Each experiment is repeated ten times under the same conditions, and the average value is used as the final result to reduce the random effects of the experiment.

5.2. Experimental Parameter Setting

5.2.1. Reward Factor

The cumulative rewards generated by three different learning rates, $\delta = 0.001$, $\delta = 0.015$, and $\delta = 0.01$ [38], to verify the feasibility analysis of the DSSO algorithm during the learning process with the number of iterations step = 1000 [22] are shown in Figure 5.

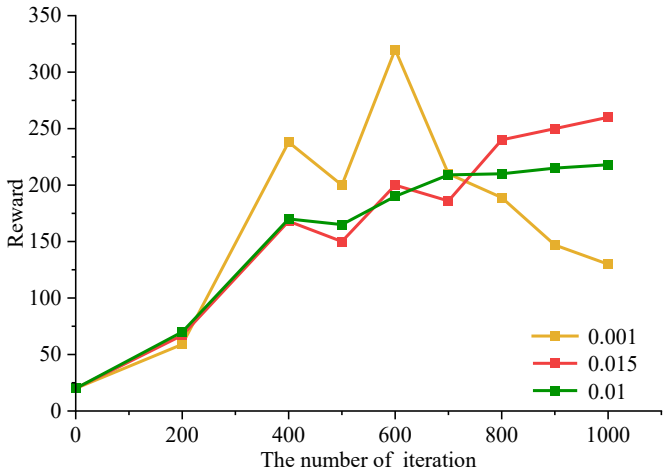

Figure 5. Convergence at different learning rates.

The reward variation caused by different learning rate δ is illustrated in Figure 5. The DSSO algorithm can be converged in $\delta = 0.001$, $\delta = 0.015$, and $\delta = 0.01$. When $\delta = 0.001$, the reward value converges most slowly among the three, which leads to low optimization efficiency; $\delta = 0.015$ the cumulative reward converges more slowly and shows a dangerous trend; $\delta = 0.01$ the reward value curve stability is optimal and connects to the local optimum stage when the number of iterations exceeds 648; therefore, this experiment selects $\delta = 0.01$ as the learning rate of the DSSO algorithm.

5.2.2. Weight Factors

The effects of different weight factors on task time latency and energy consumption are shown in Figure 6. When the vehicle D_i has sufficient computing power, it will pay more attention to shortening the task average latency when processing the task; when the vehicle D_i has insufficient computing power, it will pay more attention to reducing energy consumption.

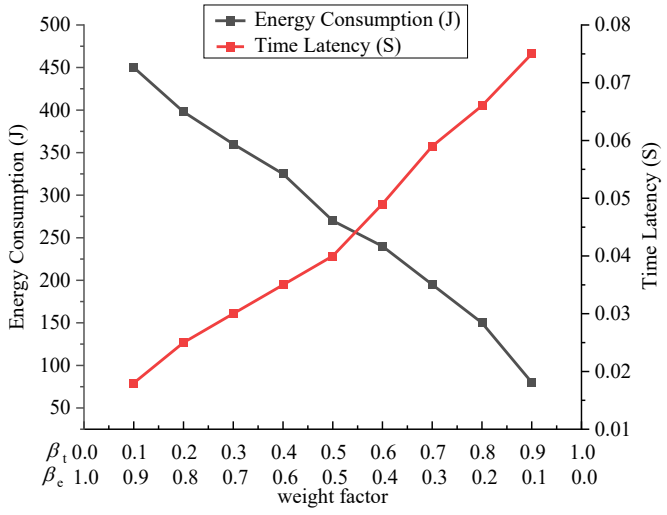

Figure 6. Impact of different weight factors on task time latency and energy consumption.

The relationship of energy consumption and time latency on different weight factors is demonstrated in Figure 6. When $\beta_t = 0.9, \beta_e = 0.1$, the task time latency of the tasks to be offloaded is slight, and the energy consumption is high. While $\beta_t = 0.1, \beta_e = 0.9$, the time latency of the task to be unloaded is max, and the energy consumption is minor. When $\beta_t = 0.55, \beta_e = 0.45$ the reward value is the highest, indicating that the offloading cost of

the vehicular tasks under edge computing is the lowest. Therefore, $\beta_t = 0.55, \beta_e = 0.45$ is selected as the weight factor for the DSSO algorithm in this experiment.

5.2.3. Evaluation Metrics

To evaluate the feasibility of the proposed algorithm DSSO, energy consumption, average delay, and packet loss rate of task offloading are selected as performance metrics.

1. Energy consumption: denotes the sum of the energy consumption of tasks on the vehicle terminals and the slices that offload the tasks to the edge server.
2. Time latency: expresses the sum of task execution delay, transmission delay, and waiting delay.
3. The ratio of dropped tasks: represents the drop ratio of in-vehicle task offloading, and this metric evaluates the completion efficiency of task offloading.

5.3. Experimental Results and Analysis

To verify the performance of the DSSO proposed in this paper, DSSO is compared with LOCAL, MINCO [39], and DJROM [40] for experiments.

1. Impact of input data size on algorithm performance

To evaluate the impact of task input data size on algorithm performance, this group of experiments set the task deadline to 1 s and the task input data size to 16 MB~48 MB. The impact of task input data size on the energy consumption generated by the four algorithms is shown in Figure 7.

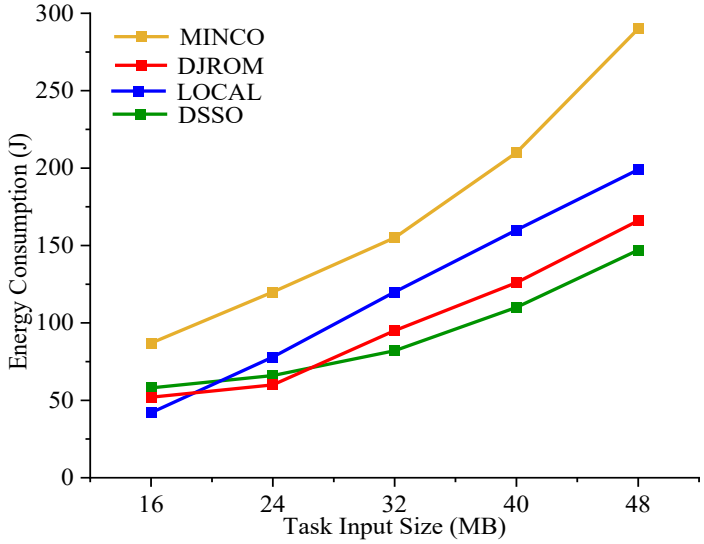

Figure 7. Impact of task input data size on energy consumption.

As can be seen from Figure 7, as the amount of in-vehicle task input data size increases, the energy consumption grows for all four algorithms; among them, the energy consumption value of MINCO is large, the energy consumption value of LOCAL algorithm is in the middle, and the energy consumption values of DJROM algorithm and DSSO algorithm are relatively low. This is because the MINCO algorithm reduces the task offloading delay by evaluating the service priority and data flow characteristics but does not restrict the energy consumption too much; the LOCAL algorithm only considers the available resources of the vehicle itself to process the task, and as the task input data volume increases, the available resources are not enough to support the task offloading, and the energy consumption value of LOCAL algorithm increases steeply; DJROM algorithm maximizes the number of computational bits by reducing the uplink power, which minimizes the energy consumption of task offloading; DSSO algorithm dynamically selects the optimal offloading strategy for the in-vehicle tasks by transforming the offloading problem of the in-vehicle task into a

Markov decision process to solve the optimal slicing result problem, using the Dueling DQN method in DRL. The DSSO algorithm highlights the advantage of reducing the energy consumption value compared with the DJROM algorithm when the task input data volume is large.

The impact of the time latency of four algorithms with different task input data sizes is given in Figure 8. It can be observed from the chart that as the task input data size increases, the time latency of all algorithms shows an upward trend. The DJROM algorithm achieves higher rates by reducing the power of the uplink, resulting in the highest time latency for tasks. The LOCAL algorithm experiences a sharp increase in the time latency of processing tasks when the task input data size reaches a particular value. The MINCO and DSSO algorithms have relatively lower time latency values. However, the DSSO algorithm has a more substantial advantage over the MINCO algorithm when facing complex network environments with enormous task offloading sizes.

The point-fold line chart of four offloading algorithms on the ratio of dropped tasks is outlined in Figure 9. The result has shown that the packet loss rate of LOCAL remains consistent when the task is offloaded, and the DSSO algorithm has the lowest packet loss rate when offloading tasks compared to MINCO and DJROM, i.e., the highest task offloading completion rate. That is because DSSO uses the loss function and the experience playback mechanism in the neural network to obtain the optimal result of slicing after mapping the true values through the decentralized operation of Dueling Network and then updates the uninstallation strategy by comparing the magnitude of the reward value, and this significantly enhances the stability of the transmission of uninstallation data.

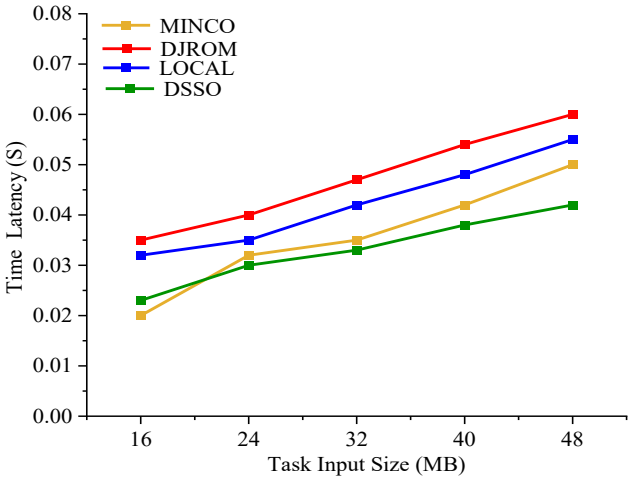

Figure 8. Impact of task input data size on time latency.

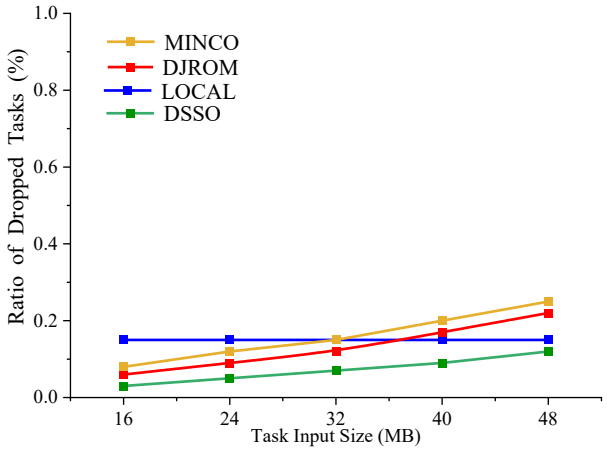

Figure 9. Impact of task input data size on the ratio of dropped tasks.

2. Impact of task deadline on algorithm performance

To evaluate the effect of task deadline on algorithm performance, the average amount of task input data set for this group of experiments is 32 MB, and the task completion deadline varies from 0.6 s to 1.4 s. The energy consumption of different offloading algorithms with different task deadlines is outlined in Figure 10.

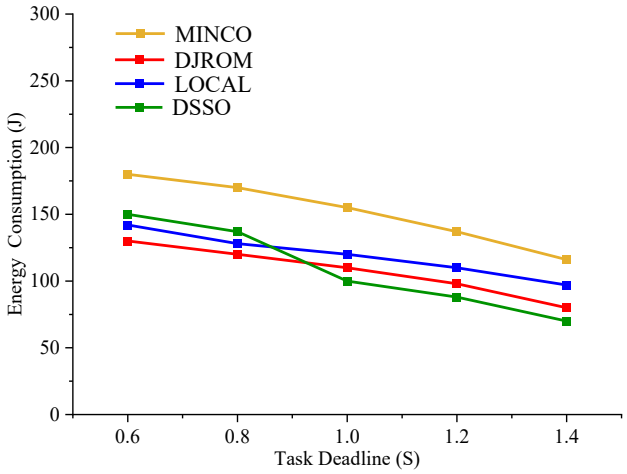

Figure 10. Impact of task deadline on energy consumption.

As can be seen from Figure 10, the energy consumption of the four algorithms decreases to varying degrees as the task deadline increases. Among them, the DSSO algorithm significantly reduces energy consumption as the task deadline increases. When the task deadline is 1.4 s compared with the LOCAL, MINCO, and DJROM algorithms, the energy consumption is reduced by 37.76%, 27.83%, and 12.15%, respectively. This is because the DSSO algorithm uses the Dueling Network in the Dueling DQN algorithm to train parameters, which only considers the advantage function that affects the offloading action, thereby reducing the energy consumption of task offloading.

The effect of task deadlines on the time latency of the four algorithms is illustrated in Figure 11. From this figure, it can be seen that the time latency of all four algorithms increases as the task deadline increases. Among them, DSSO offloads the task to the network slice in the edge server through the collaborative interaction between the in-vehicle terminal and the edge server, which dramatically reduces the time delay of the task by 24.32%, 36.67%, and 6.67% compared with LOCAL, DJROM, and MINCO when the task deadline is 1.2 s.

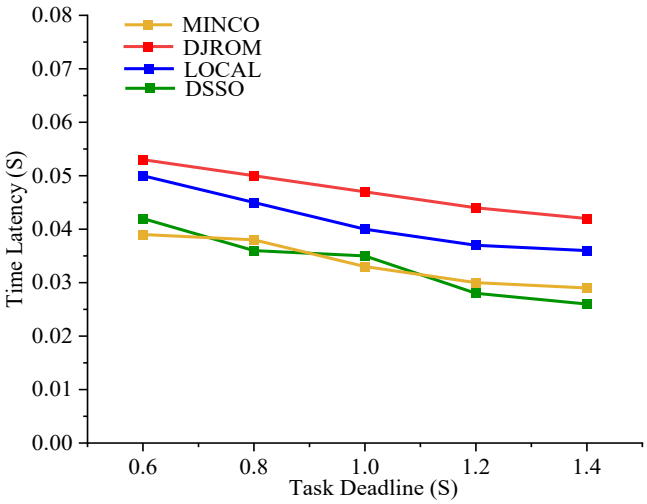

Figure 11. Impact of task deadline on time latency.

As can be seen from Figure 12, without considering energy consumption and time latency, the packet loss rate of all four algorithms exhibits a significant decreasing trend with the increasing task deadline. However, the DSSO algorithm shows the most obvious advantage because it dynamically adjusts the slicing through continuous updates of neural network parameters, thereby obtaining an optimal offloading strategy within the effective range.

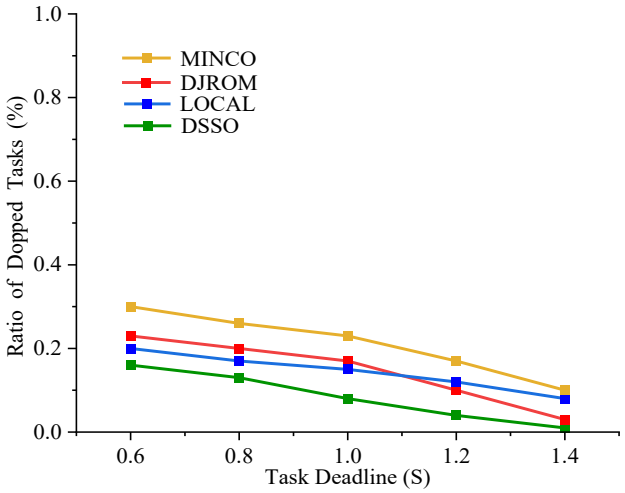

Figure 12. Impact of task deadline on the ratio of dropped tasks.

6. Conclusions

In this study, the offloading problem of relationship-intensive and delay-sensitive vehicular tasks in MEC is investigated. The distributed offloading algorithm was designed to enable devices to make offloading decisions dispersedly, effectively solving the load imbalance problem at edge nodes. The Dynamic Selection Slicing-based Offloading Algorithm for in-vehicle Task in MEC was proposed. The offloading model of in-vehicle tasks is constructed by considering vehicle conditions, task dependencies, and available resources. The DRL approach was used to dynamically select the optimal task offloading strategy. The experimental results showed that the DSSO algorithm achieved the optimization goal when facing massive service requests and large data scales and performed better than similar algorithms in energy consumption, time latency, and task completion rate. However, due to the link disconnection caused by the high-speed movement of vehicles and the possible security hazards during data computation offloading. In the future, the security issues of offloading to in-vehicle task collaboration computing under the edge computing environment will be investigated.

Author Contributions: Conceptualization, L.H. and Y.B.; methodology, L.H.; software, Y.B.; validation, Y.L., Y.B. and S.Z.; formal analysis, L.H.; investigation, Y.B.; resources, S.Z.; data curation, S.Z.; writing—original draft preparation, Y.B.; writing—review and editing, L.H.; visualization, Y.L.; supervision, S.Z.; project administration, S.Z.; funding acquisition, Y.B. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the National Natural Science Foundation (NSF) under grants (No. 61802353), the Natural Science Foundation of Henan Province (No. 202300410505), and the project of Science and Technology of Henan Province (No. 192102210270).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Saeik, F.; Avgeris, M.; Spatharakis, D.; Santi, N.; Dechouniotis, D.; Violos, J.; Leivadeas, A.; Athanasopoulos, N.; Mitton, N.; Papavassiliou, S. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Comput. Netw.* **2021**, *195*, 108177. [CrossRef]
2. Singh, A.; Satapathy, S.C.; Roy, A.; Gutub, A. Ai-based mobile edge computing for IoT: Applications, challenges, and future scope. *Arab. J. Sci. Eng.* **2022**, *47*, 9801–9831. [CrossRef]

3. Ma, Z. Communication Resource Allocation Strategy of Internet of Vehicles Based on MEC. *J. Inf. Process. Syst.* **2022**, *18*, 389–401.
4. Garcia, M.H.C.; Molina-Galan, A.; Boban, M.; Gozalvez, J.; Coll-Perales, B.; Şahin, T.; Kousaridas, A. A tutorial on 5G NR V2X communications. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1972–2026. [CrossRef]
5. Zhang, X.; Zhang, J.; Liu, Z.; Cui, Q.; Tao, X.; Wang, S. MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3296–3309. [CrossRef]
6. Tang, D.; Zhang, X.; Tao, X. Delay-optimal temporal-spatial computation offloading schemes for vehicular edge computing systems. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–6.
7. Kim, J. Service Aware Orchestration for Dynamic Network Slicing in 5G Networks. In *Advances in Computer Science and Ubiquitous Computing: CSA-CUTE 17*; Springer: Singapore, 2018; pp. 1397–1402.
8. Wijethilaka, S.; Liyanage, M. Survey on network slicing for Internet of Things realization in 5G networks. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 957–994. [CrossRef]
9. Jiang, W.; Zhan, Y.; Zeng, G.; Lu, J. Probabilistic-forecasting-based admission control for network slicing in software-defined networks. *IEEE Internet Things J.* **2022**, *9*, 14030–14047. [CrossRef]
10. Gonçalves, D.M.; Puliafito, C.; Mingozzi, E.; Bittencourt, L.F.; Madeira, E.R. End-to-end network slicing in vehicular clouds using the MobFogSim simulator. *Ad Hoc Netw.* **2023**, *141*, 103096. [CrossRef]
11. Saleem, M.A.; Mahmood, K.; Kumari, S. Comments on "AKM-IoV: Authenticated key management protocol in fog computing-based internet of vehicles deployment". *IEEE Internet Things J.* **2020**, *7*, 4671–4675. [CrossRef]
12. Yazdinejad, A.; Parizi, R.M.; Dehghantanha, A.; Zhang, Q.; Choo, K.K.R. An energy-efficient SDN controller architecture for IoT networks with blockchain-based security. *IEEE Trans. Serv. Comput.* **2020**, *13*, 625–638. [CrossRef]
13. Zhou, S.; Huang, H.; Chen, W.; Zhou, P.; Zheng, Z.; Guo, S. Pirate: A blockchain-based secure framework of distributed machine learning in 5g networks. *IEEE Netw.* **2020**, *34*, 84–91. [CrossRef]
14. Bondan, L.; Wauter, T.; Volckaert, B.; De Turck, F.; Granville, L.Z. NFV Anomaly Detection: Case Study through a Security Module. *IEEE Commun. Mag.* **2022**, *60*, 18–24. [CrossRef]
15. Hussain, M.; Shah, N.; Amin, R.; Alshamrani, S.S.; Alotaibi, A.; Raza, S.M. Software-defined networking: Categories, analysis, and future directions. *Sensors* **2022**, *22*, 5551. [CrossRef] [PubMed]
16. Chen, C.; Zeng, Y.; Li, H. A multi-hop task offloading decision model in MEC-enabled internet of vehicles. *IEEE Internet Things J.* **2022**, *10*, 3215–3230. [CrossRef]
17. Yu, J.; Lu, L.Y.; Li, X. Edge-cloud collaborative task offloading mechanism based on DDQN in vehicular networks. *Comput. Eng.* **2022**, *48*, 156–164.
18. Abinaya, A.B.; Karthikeyan, G. Data Offloading in the Internet of Vehicles Using a Hybrid Optimization Technique. *Intell. Autom. Soft Comput.* **2022**, *34*, 325–338. [CrossRef]
19. Cao, B.; Fan, S.; Zhao, J.; Tian, S.; Zheng, Z.; Yan, Y.; Yang, P. Large-scale many-objective deployment optimization of edge servers. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3841–3849. [CrossRef]
20. Bi, S.; Zhang, Y.J. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 4177–4190. [CrossRef]
21. Tang, M.; Wong, V.W.S. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Trans. Mob. Comput.* **2020**, *21*, 1985–1997. [CrossRef]
22. Cheng, Z.; Wang, Q.; Li, Z.; Rudolph, G. Computation offloading and resource allocation for mobile edge computing. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 2735–2740.
23. Kim, K.; Lynskey, J.; Kang, S.; Hong, C.S. Prediction based sub-task offloading in mobile edge computing. In Proceedings of the 2019 International Conference on Information Networking (ICOIN), Kuala Lumpur, Malaysia, 9–11 January 2019; pp. 448–452.
24. Li, C.; Tang, J.; Zhang, Y.; Yan, X.; Luo, Y. Energy efficient computation offloading for nonorthogonal multiple access assisted mobile edge computing with energy harvesting devices. *Comput. Netw.* **2019**, *164*, 106890. [CrossRef]
25. Poularakis, K.; Llorca, J.; Tulino, A.M.; Taylor, I.; Tassiulas, L. Joint service placement and request routing in multi-cell mobile edge computing networks. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 10–18.
26. Chen, X.; Zhang, H.; Wu, C.; Mao, S.; Ji, Y.; Bennis, M. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. *IEEE Internet Things J.* **2018**, *6*, 4005–4018. [CrossRef]
27. Van Huynh, N.; Hoang, D.T.; Nguyen, D.N.; Dutkiewicz, E. Optimal and fast real-time resource slicing with deep dueling neural networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1455–1470. [CrossRef]
28. Al-Khatib, A.A.; Khelil, A. Priority-and reservation-based slicing for future vehicular networks. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization, Ghent, Belgium, 29 June–3 July 2020; pp. 36–42.
29. Peng, H.; Shen, X. Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 131–141. [CrossRef]
30. Mlika, Z.; Cherkaoui, S. Network slicing with MEC and deep reinforcement learning for the Internet of Vehicles. *IEEE Netw.* **2021**, *35*, 132–138. [CrossRef]

31. Nassar, A.; Yilmaz, Y. Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities. *IEEE Internet Things J.* **2021**, *9*, 222–235. [CrossRef]
32. Li, M.; Gao, J.; Zhao, L.; Shen, X. Deep reinforcement learning for collaborative edge computing in vehicular networks. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 1122–1135. [CrossRef]
33. Huang, L.; Bi, S.; Zhang YJ, A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Trans. Mob. Comput.* **2019**, *19*, 2581–2593. [CrossRef]
34. Ajayi, J.; Di Maio, A.; Braun, T.; Xenakis, D. An Online Multi-dimensional Knapsack Approach for Slice Admission Control. In Proceedings of the 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023; pp. 152–157.
35. Eftimie, A.; Borcoci, E. SDN controller implementation using OpenDaylight: Experiments. In Proceedings of the 2020 13th International Conference on Communications, Bucharest, Romania, 8–20 June 2020; pp. 477–481.
36. Salvat, J.X.; Zanzi, L.; Garcia-Saavedra, A.; Sciancalepore, V.; Costa-Perez, X. Overbooking network slices through yield-driven end-to-end orchestration. In Proceedings of the 14th international conference on emerging networking experiments and technologies, Heraklion, Greece, 4–7 December 2018; pp. 353–365.
37. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
38. Lu, J.; Jiang, J.; Balasubramanian, V.; Khosravi, M.R.; Xu, X. Deep reinforcement learning-based multi-objective edge server placement in Internet of Vehicles. *Comput. Commun.* **2022**, *187*, 172–180. [CrossRef]
39. Khoramnejad, F.; Erol-Kantarci, M. On joint offloading and resource allocation: A double deep q-network approach. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 1126–1141. [CrossRef]
40. Lu, J.; Shi, Z.; Wang, Y.; Pan, C.; Zhang, S. Multi-index evaluation learning-based computation offloading optimization for power internet of things. *Phys. Commun.* **2023**, *56*, 101949. [CrossRef]