Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3213311, 11 pages
https://doi.org/10.1155/2022/3213311

Research Article

Time-Driven Scheduling Based on Reinforcement Learning for Reasoning Tasks in Vehicle Edge Computing

Bing Lin,[1] **Qiaoxin Chen**,[1] **and Yu Lu**[2]

[1]*College of Physics and Energy, Fujian Normal University, Fuzhou 350117, China*
[2]*Concord University College, Fujian Normal University, Fuzhou 350117, China*

Correspondence should be addressed to Yu Lu; fzluyu@163.com

Received 19 October 2021; Revised 13 January 2022; Accepted 22 January 2022; Published 23 February 2022

Academic Editor: Chi-Hua Chen

Significant challenges for reasoning tasks scheduling remain, including the selection of an optimal tasks-servers solution from the possible numerous combinations, due to the heterogeneous resources in edge environments and the complicated data dependencies in reasoning tasks. In this study, a time-driven scheduling strategy based on reinforcement learning (RL) for reasoning tasks in vehicle edge computing is designed. Firstly, the reasoning process of vehicle applications is abstracted as a model based on directed acyclic graphs. Secondly, the execution order of subtasks is defined according to the priority evaluation method. Finally, the optimal tasks-servers scheduling solution is chosen by Deep Q-learning (DQN). The extensive simulation experiments show that the proposed scheduling strategy can effectively reduce the completion delay of reasoning tasks. It performs better in algorithm convergence and runtime compared with the classic algorithms.

1. Introduction

In recent years, Internet of Vehicles (IoV) has become a research hotspot for the Intelligent Transportation System (ITS) [1]. The autonomous driving of IoV not only improves the driving safety, but also solves the problem of traffic inefficiency and lane congestion. It is a challenge for the autonomous driving to complete the target application under strict time constraint and restricted computing resources. The current work for autonomous driving mostly focuses on how to design the specific functions, such as traffic recognition, into reasoning tasks [2, 3]. Less attention is paid to how to schedule these reasoning tasks of autonomous driving to the appropriate computing nodes with low latency. Fortunately, IoV in Mobile Edge Computing (MEC) could schedule the real-time tasks from vehicles to the Road Side Units (RSU) with powerful computing resources, alleviating the task execution delay. Besides, a reasonable scheduling for reasoning tasks in MEC can effectively reduce both the execution latency of tasks and the workload of vehicles [4–11]. However, due to the heterogeneous resources in edge environments and the complicated data dependencies in reasoning tasks, significant challenges for reasoning tasks scheduling remain, including the selection of an optimal tasks-servers solution from the possible numerous combinations.

Existing studies are mainly done subject to task scheduling and task coordination through heuristic algorithms [12–15], such as Particle Swarm Optimization (PSO), Colony Algorithm (CA), and Genetic Algorithm (GA). Although these works could obtain the feasible solutions while satisfying different constraints, they fail to predict the deviation between the feasible and optimal solutions in advance, which makes their solutions easily fall into the local optimum. Several studies have been devoted to task scheduling using reinforcement learning (RL) algorithm [16–26], which can not only correct the deviation between the feasible and optimal solutions, but also accelerate the convergence of perfect results. Specifically, Lin et al. [23] proposed a time-driven scheduling strategy based on Q-learning algorithm for reasoning tasks of autonomous driving in IoV. The experimental results demonstrated that the performance of RL algorithms based on simulated annealing was better than other classic algorithms. This work

is instructive for our work. Zhao et al. [20] put forward a distribution scheduling algorithm based on DQN to achieve the best balance between latency, computational rate, and energy consumption, for an edge access network of IoV. They prioritized the tasks of different vehicles according to the analytic hierarchy process (AHP). The experimental results showed that the average task processing delay of the proposed method could effectively improve the task offload efficiency. However, the priority between tasks has not been scientifically calculated and weighted, but only evaluated by experts based on their experience. The current work [27, 28] for priority evaluation is mostly subjective by experts. There are great achievements in multivehicle task collaborative scheduling [5, 7, 11, 20]. However, the time-driven scheduling for single-vehicle reasoning tasks with data dependencies is still an open issue.

In response to this issue, two research questions are considered: (1) how to design a model for the reasoning tasks with data dependencies to evaluate the latency caused by task execution and data transmission? (2) How to develop an efficient and reliable scheduling strategy to reduce the latency during vehicle driving? To solve the above questions, we design time-driven scheduling based on RL for reasoning tasks in vehicle edge computing, which considers the differences of heterogeneous real-time reasoning tasks and optimizes the completion latency of reasoning tasks.

The main contributions of this paper are concluded as follows:

(1) A latency model is designed for reasoning tasks with data dependencies, which considers the latency caused by task execution and data transmission

(2) The scheduling for reasoning tasks in MEC is defined as a Markov Decision Process (MDP), which models the scheduling strategy for a reasoning task as the state, the resource allocation decision for each subtask as the action, and the completion latency of a reasoning task as the reward

(3) A time-driven scheduling strategy based on DQN is designed to explore an optimal tasks-servers solution from the possible numerous combinations in vehicle edge computing

The remaining part of the paper proceeds as follows. We review the related work in Section 2. Section 3 introduces the problem definitions of reasoning tasks scheduling. Section 4 describes the proposed reasoning tasks scheduling strategy in detail. Section 5 conducts the comparative experiments and analyzes the performance of the proposed strategy. Finally, Section 6 summarizes the work of this paper and looks forwards to the future research directions.

2. Related Work

Task scheduling in MEC has been extensively studied [4–10]. In general, task scheduling approaches mainly include the methods based on heuristic algorithms [12–15] and reinforcement learning [16–26].

2.1. Methods Based on Heuristic Algorithms. Xie et al. [12] proposed a novel Directional and Non-local-Convergent Particle Swarm Optimization (DNCPSO) to address workflow scheduling in cloud-edge environment, which could reduce the make span and cost dramatically and work well for task scheduling in complex applications. Wu et al. [13] studied how to dynamically partition a given application effectively into local and remote parts while reducing the total cost in cloud-edge environment. They proposed a Min-Cost Offloading Partitioning (MCOP) algorithm, which could significantly reduce the execution time and energy consumption by optimally distributing tasks between mobile devices and servers. Lin et al. [15] proposed a linear-time rescheduling algorithm for the task migration in MCC environment. The algorithm started from a minimal-delay scheduling solution and subsequently performed energy reduction by migrating tasks among the local cores and the cloud.

The methods based on heuristic algorithms can easily fall into the local optimal solution, which usually fails to get a good result. Moreover, the time required to process reasoning tasks of IoV is usually strict. The methods based on heuristic algorithms are not suitable for such problem due to their long algorithm execution time.

2.2. Methods Based on Reinforcement Learning. To adapt the scheduling strategies for dynamic scenarios, Deep Reinforcement Learning (DRL) has been widely applied to the task scheduling problems in MEC systems in recent years.

Chen et al. [16] designed a double DQN-based computation scheduling policy for a virtual MEC system. Numerical experiments showed that their proposed policy could achieve a significant improvement in computation scheduling performance. Xiong et al. [17] proposed an improved DQN algorithm to minimize the long-term weighted sum of average completion time of jobs and average number of requested resources in IoT edge computing system. Simulation results showed that the proposed algorithm has a better performance than the original DQN algorithm. Wang et al. [18] proposed a new DRL-based scheduling framework to address the challenges of task dependency and adapting to dynamic scenarios in the MEC system. The proposed DRL solution could automatically discover the common patterns behind various applications so as to infer an optimal scheduling policy in different scenarios. Rjoub et al. [21, 26] proposed four deep and RL-based scheduling approaches to automate the process of scheduling large-scale workloads onto cloud computing resources, while reducing both the resource consumption and task waiting time. These approaches derived an appropriate task scheduling mechanism that could minimize both tasks' execution delay and cloud resources utilization. Qi et al. [22] firstly proposed a multitask DRL approach for scalable parallel task scheduling (MDTS) in IoV. For avoiding the curse of dimensionality when coping with complex parallel computing environments and jobs with diverse properties, they extended the action selection in DRL to a multitask decision, where the output branches of

multitask learning were fine-matched to parallel scheduling tasks. Huang et al. [24] proposed a DRL-based Online Offloading (DROO) framework to optimally adapt task scheduling decisions and wireless resource allocations to the time-varying wireless channel conditions in a wireless powered MEC network. Numerical results showed that the proposed framework could achieve near-optimal performance while significantly decreasing the computation time.

RL-based methods mostly assume that the scheduling problem is a learning task. Through preliminary training, an effective scheduling policy for the task can be quickly formed by a reasonable designed RL algorithm. Note that current work for IoV mostly focuses on multivehicle collaborative scheduling, but the time-driven scheduling for single-vehicle reasoning tasks with data dependencies is still an open issue.

3. Problem definition

Table 1 shows the notations used in this paper.

Figure 1 gives an example of reasoning tasks scheduling in vehicle edge computing. This example considers autonomous driving reasoning system [2, 3], which consists of applications such as emergency rule inference engine and security operations. The user equipment (UE) makes scheduling decision for those applications according to the status in edge environments and application profiles; thus some of them are executed locally on the vehicle (i.e., UE) while others are scheduled to the edge by wireless channels. In this work, we consider that the edge environment is composed of m RSUs providing resources including computing, communications, and storage to UE in each time-slot, which is expressed as $\mathbf{F} = \{f_j \mid 1 < j < m\}$. The computation capacities of vehicle and RSU are denoted as f_{vehicle} and f_{RSU}.

In time-slot i, a reasoning task can be expressed as a directed acyclic graph (DAG) $\mathbf{G} = \langle \mathbf{N}, \mathbf{E} \rangle$ as in Figure 2, where $\mathbf{N} = \{n_i \mid 1 < i < z\}$ is a set of z subtasks and $\mathbf{E} = \{\mathbf{e_{u,k}} \mid 1 \le u, k \le z, u \ne k\}$ is the data dependencies between subtasks. The data dependency $\mathbf{e_{u,k}}$ indicates that there is a directed arc between subtasks n_u and n_k, and task n_k cannot start until task n_u has been completed. The set of direct precursors of task n_k can be expressed as $\mathbf{R_k} = \{n_k \mid 1 \le k \le z, k \ne j, \mathbf{e_{u,k}}\}$. A task cannot be executed until all of its direct precursors are completed.

In vehicle edge computing, a subtask in the reasoning task can be either offloaded to the edge or executed locally on the vehicle. If the offloading occurs for a reasoning task, the process delay will be related to the subtask profile and the environment state. The subtask profile includes the required CPU cycles for running the subtasks c_i, data size of the subtasks data_i, and the tolerable delay of the subtasks t_d^i. Besides, the environment state contains the transmission rate of wireless channel $v_{\text{transmission}}^{m_i}$. Therefore, the transmission latency $t_{\text{transmission}}^i$ for subtasks executing on edge nodes can be calculated as (1) and (2). If subtasks are executed locally on the vehicle, there is only execution latency on the user equipment, which can be obtained by t_{vehicle}^i.

$$t_{\text{transmission}}^i = \frac{\text{data}_i}{v_{\text{transmission}}^{m_i}}, \tag{1}$$

$$t_{\text{RSU}}^i = \frac{c_i}{f_{\text{RSU}}}, t_{\text{vehicle}}^i = \frac{c_i}{f_{\text{vehicle}}}. \tag{2}$$

The scheduling plan for a reasoning task is denoted as distribution relationship matrix as (3). If $a_{x,y} = 1$, it means that subtask n_x is offloaded to edge node m_y; otherwise, it means that subtask n_x is executed locally. When the edge node is running normally, the execution latency of the reasoning tasks can be expressed by (4), where $t_{process}$ means the processing latency of the reasoning tasks. If there is no available edge node in the edge environment, all subtasks will be executed serially on the vehicle, where m_i is set to 0. When the worst scheduling occurs, the completion latency of the reasoning tasks is described as in equation (5):

$$\mathbf{A}_{\mathbf{m_i} \times \mathbf{z_i}} = \begin{Bmatrix} 1 & \cdots & a_{1,z_i} \\ \cdots & \cdots & \cdots \\ 0 & \cdots & a_{m_i,z_i} \end{Bmatrix}, \tag{3}$$

$$t_{\text{all}} = t_{\text{process}} + \text{MAX}\left(t_{\text{transmission}}^i\right), \tag{4}$$

$$t_{\text{worst}} = \sum_{i=0}^{z} t_{\text{vehicle}}^i. \tag{5}$$

To make better use of computing resources in different edge environments, we assume that edge nodes should satisfy the following processing principles:

(1) A subtask is processed by only one corresponding edge node, which is formally defined as (6).

(2) After all subtasks are assigned to the corresponding edge nodes, the edge nodes begin to process the subtasks.

(3) The subtasks on different edge nodes without data dependencies can be processed in parallel.

(4) The subtasks on the same edge node are processed according to the data dependencies. Otherwise, they are processed according to their corresponding priorities.

(5) The tolerable delay of the subtasks on the edge node is not greater than the execution latency of the corresponding edge node, which is formally defined as (7):

$$\sum_{y=1}^{z_i} a_{x,y} \le 1, \tag{6}$$

$$\begin{aligned} t_{\text{RSU}}^i &\le t_d^i, \\ t_{\text{vehicle}}^i &\le t_d^i. \end{aligned} \tag{7}$$

The reasoning task scheduling discussed in this paper can be summarized as follows: in various time-slots, a reasoning task on a vehicle can be decomposed into several

Table 1: Notations and descriptions.

Notation	Description
$\mathbf{G}$, $\mathbf{N}$, $\mathbf{E}$	A reasoning task, a set of subtasks, and the data dependencies between subtasks
$\mathbf{A}_{\mathbf{m_i \times z_i}}$	Computational scheduling plan for n tasks
c_i, data_i, t_d^i	Task size, computation intensity, tolerable delay of subtask n_i
t^i_{vehicle}, t_{process}	Computation latency in vehicle and RSU
f_{vehicle}, f_{RSU}	Local computing capacity and scheduling capacity
$t^i_{\text{transmission}}$	Transmission latency of subtasks
t_{all}	Completion latency of a reasoning task
s_t, a_t, r_t	State, action, and reward of an MDP at time step t
α, γ	Parametrized policy and value function for computation scheduling

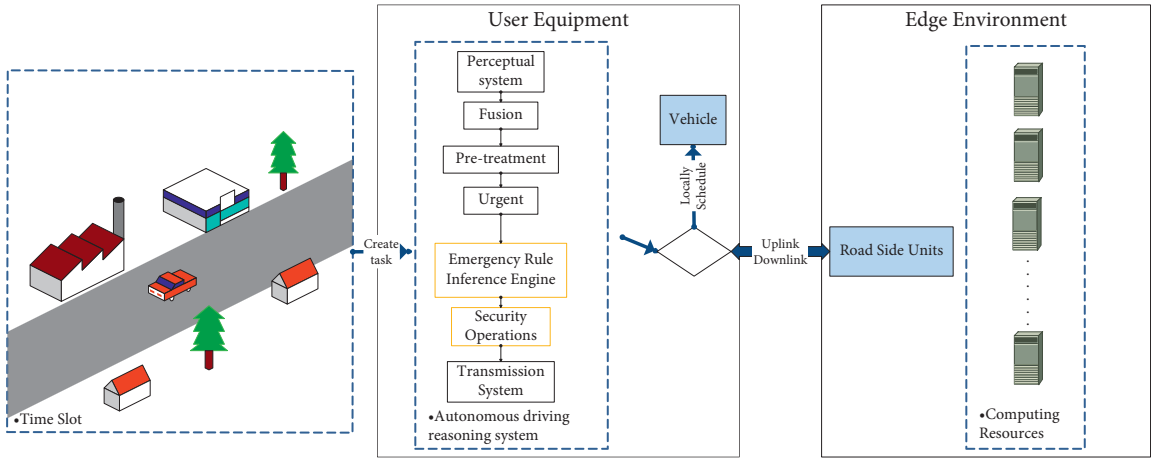

Figure 1: An example of reasoning tasks scheduling in vehicle edge computing.

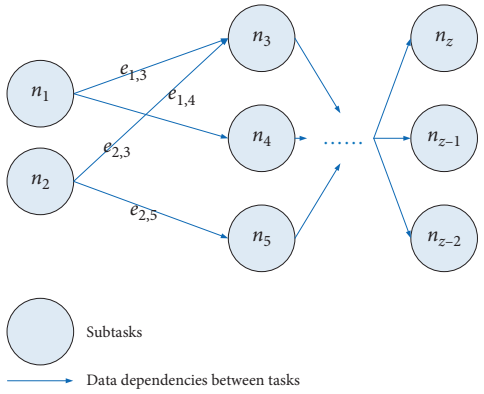

Figure 2: A reasoning task.

subtasks, and these subtasks can be reasonably scheduled to the edge nodes for processing based on a specific scheduling algorithm. The scheduling algorithm proposed in this paper can minimize the execution latency of reasoning tasks, which can be expressed as

$$\text{Minimize } t_{\text{all}}, t_{\text{all}} \le t_{\text{worst}}. \tag{8}$$

4. Algorithm Design

In this section, we first describe the priority evaluation for subtasks in a reasoning task, which is employed to determine the order of execution for the subtasks without data dependencies. And then give an overview of our proposed scheduling algorithm. Finally, we introduce the implementation of the scheduling algorithm in detail.

4.1. Priority Evaluation for Each Subtask. It is difficult to estimate the execution time of a reasoning task that defines the execution sequence of subtasks. Fuzzy analytic hierarchy process (FAHP) [27–29] is usually employed to analyze multiobjective problems, which decomposes the problem hierarchically according to its feature and overall goal, forming a bottom-up gradient hierarchy. In this work, FAHP is commonly used to measure the subtask weight, which can determine the order of execution for the subtasks without data dependencies. Each subtask weight is modified by calculating the information entropy of objective factors (i.e., each subtask's own parameters) [30, 31]. The pseudocode of the priority evaluation for each subtask is described as follows:

$$p_{i,j} = \begin{cases} 0, s_i < s_j, \\ 0.5, s_i = s_j, \\ 1, s_i > s_j, \end{cases} \tag{9}$$

$$\mathbf{R} = \left(r_{i,j}\right)_{\alpha \cdot \alpha}, r_{i,j} = r_{i,k} - r_{j,k} + 0.5, \tag{10}$$

Input: computational complexity a_1, the amount of data a_2, the tolerable delay a_3
Output: the priority of subtask z
(1) Sort subtask's factor according to equation (9) and construct matrix $\boldsymbol{P}$
(2) **for** $i \longleftarrow 0$ to maximum rows of **P do**
(3) $r_i = 0$
(4) **for** $j \longleftarrow 0$ to maximum columns of **P do**
(5) $r_i = r_i + p_{i,j}$
(6) **end for**
(7) **end for**
(8) r_i are transformed through equation (12) to obtain **R**
(9) **for** i $\longleftarrow 0$ to maximum rows of **R do**
(10) $w_i = 0$
(11) **for** $j \longleftarrow 0$ to maximum columns of **R do**
(12) update w_i via equation (13)
(13) **end for**
(14) **end for**
(15) calculate the information entropy δ_i via equations (14) and (15)
(16) obtain w_i' via (16)
(17) $z = w_1' \cdot a_1 + w_2' \cdot a_2 + w_3' \cdot a_3$

Algorithm 1: Priority evaluation for each subtask.

$$r_i = \sum_{k=1}^{\alpha} r_{i,k}, \quad i = 1, 2, \ldots, \alpha, \tag{11}$$

$$r_{ij} = \frac{r_i - r_j}{2 \cdot \alpha} + 0.5, i = 1, 2, \ldots, \alpha, \tag{12}$$

$$w_i = \frac{\sum_{k=1}^{\alpha} r_{i,k}}{\sum_{i=1}^{\alpha}\sum_{j=1}^{\alpha} r_{i,j}} = \frac{2}{\alpha^2}\sum_{k=1}^{\alpha} r_{i,k}, i = 1, 2, \ldots, \alpha, \tag{13}$$

$$\gamma_{ij} = \frac{\max(a_i) - a_{i,j}}{\max(a_i) - \min(a_i)}, \tag{14}$$

$$\delta_i = -\frac{1}{\ln n}\sum_{j=1}^{\alpha} \beta_{ij} \ln \beta_{ij}, \quad 0 \le \delta_i \le 1, \beta_{ij} = \frac{\gamma_{ij}}{\sum_{j=1}^{\alpha} \gamma_{ij}}, \tag{15}$$

$$\begin{cases} g_i = \dfrac{1 - \delta_i}{\sum_{i=1}^{\alpha} \delta_i}, \\ w_i' = \dfrac{w_i \cdot g_i}{\sum_{i=1}^{\alpha} w_i \cdot g_i}, \end{cases} \tag{16}$$

s_i and s_j represent the relative importance of subtask factors. α and δ are the number of factors and the information entropy of them.

4.2. Scheduling Algorithm. MDP is a basic model of the RL in this paper. The scheduling algorithm can be simplified according to the MDP property, which means that the next state is only related to the current state as Figure 3. In Figure 3, each state represents a corresponding allocation strategy for real-time vehicle tasks in different edge environment and corresponds to a specific reward. Each action is calculated by the agent (neural network), and it is used to guide the current state to a better direction.

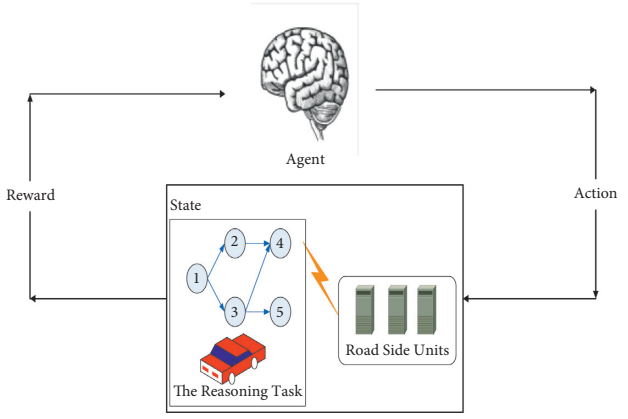

Figure 3: The MDP model of scheduling algorithm.

$$\begin{cases} Q_{k+1}(s_t, a_t, \theta_t) = Q_k(s_t, a_t, \theta_t) + \alpha_k \cdot R_k, \\ R_k = r_t + \gamma \cdot \max_{a \in a} Q(s', a', \theta_t') - Q(s_t, a_t, \theta_t), \end{cases} \tag{17}$$

The model characteristics of the discussed problem in this paper are described as follows:

(1) State space: the number of states for the feasible solutions is not constant. They can change dynamically as the change of the number of subtasks after decomposition and the changed distribution of edge nodes in various time-slots.

(2) Action space: the number of optional actions in the action space is equal to the number of subtasks. Action selection means scheduling the corresponding subtasks in current state to the specific edge nodes.

(3) Reward value: this work tries to minimize the completion latency of the reasoning task, so the reward value is set to $r_t = (1/t_{\text{all}})$.

Input: initial state, maximum number of rounds, maximum number of iterations in a single round
Output: the scheduling strategy for reasoning tasks
(1) Initialize the experience pool of constant storage space, the action-value function $Q_\theta(s_t, a_t)$ with random weight θ and the corresponding target – $Q_\theta(s_t, a_t)$
(2) **for** $i \leftarrow 0$ to Maximum number of rounds **do**
(3) $s_t \leftarrow$ initial state
(4) **for** $i \leftarrow 0$ to maximum number of iterations in a single round **do**
(5) Choose the action $a_t = \text{argmax}Q(s_t, a, \theta)$ with the largest historical reward with possibility ε, otherwise choose a random action
(6) Execute action a_t to get the next state s_{t+1} and use Algorithm 2 to calculate the reward r_t
(7) Store (s_t, s_{t+1}, r_t, a_t) in the experience pool
(8) $s_t \leftarrow s_{t+1}$
(9) Random sampling (s_j, s_{j+1}, r_j, a_j) from the experience pool
(10) Construct an error function according to equation (17), and back-propagation to update the parameters θ
(11) Update target – $Q_\theta(s_t, a_t) = Q_\theta(s_t, a_t)$ per few steps
(12) If s_{t+1} satisfies the termination state, the current iteration is ended
(13) end for
(14) end for

Algorithm 2: Scheduling algorithm.

The scheduling strategy is based on the DQN algorithm. It can be abstracted as a function fitting problem when the discrete tangent dimension of the state and action space are high. The pseudocode of our scheduling algorithm is described in Algorithm 2.where α_k and γ represent the learning rate and discount factor, respectively. s' is the state after executing the action a_t in the k_{th} iteration.a' represents the action of the largest reward in state s', and R_k represents the accumulated reward during the iterations.

4.3. Algorithm Implementation. In various time-slots, reasoning tasks and edge environments can change dynamically. These changes are summarized as follows:

(1) The topological structure of reasoning tasks and the number of nodes in edge environments
(2) The computational complexity, the datasets between subtasks, and the tolerable delay of subtasks in various environments
(3) The transmission latency and execution latency of subtasks

The algorithm implementation will calculate the completion latency of reasoning tasks in edge environments. The pseudocode of the algorithm implementation is described in Algorithm 3. Figure 4 presents the calculation process of execution latency, which includes the following steps.

Step 1: initiate the parameters of Algorithm 3, including the subtask queue Q and the set of predecessor nodes R. Next, a reasoning task is expressed as a specific directed acyclic graph.

Step 2: Q is used to sort the subtasks by the topology of the reasoning task.

Step 3: calculate the task execution time according to the specific strategy derived from Algorithm 2.

5. Simulation Experiment and Analysis

5.1. Experimental Parameter Settings. The simulation experiments are implemented with the Python 3.7 and conducted on a 64-bit Win10 system, which is configured with Inter(R) Core (TM) i7-7700HQ CPU and 16 GB RAM. Our proposed scheduling algorithm is DQN, and Q-learning algorithm [23] and GA-PSO [32] are introduced as the comparison algorithms. Based on the effects of adjusting parameters in many experiments, the corresponding parameters of DQN and Q-learning [23] are set as $\alpha = 0.005$, $\gamma = 0.9$, and $\varepsilon = 0.9$. The corresponding parameters of GA-PSO [32] are set as $w_{\max} = 0.9$, $w_{\min} = 0.4$, $C_1^{\text{start}} = 0.9$, $C_1^{\text{end}} = 0.2$, $C_2^{\text{start}} = 0.9$, and $C_2^{\text{end}} = 0.4$. In addition, the number of rounds is set as 100 and the number of iterations is set as 1000 for DQN, Q-learning, and GA-PSO, respectively.

All the algorithms try to find the optimal scheduling result with the shortest completion latency of reasoning tasks in edge environments.

UEs have different reasoning tasks with various topologies and task number, and the topological structure of reasoning tasks is shown in Figure 5. The related parameters for the vehicle edge computing environment are set according to the IEEE 802.11p [33], and other parameters are set as Table 2.

5.2. Analysis of Results. Table 3 shows the completion latency of different reasoning tasks in various edge environments with our proposed scheduling algorithm, where m and n denote the number of edge nodes and subtasks in each experiment. Note that $n = 6$ corresponds to the "Topology I," $n = 9$ corresponds to the "Topology II," and $n = 12$ corresponds to the 'Topology III' in Figure 5. Each grid in Table 3 corresponds to an experiment with different reasoning tasks with specific topologies and different edge nodes in edge environments. In addition, the execution order of subtasks is

Input: m_i, z_i, G_i $A_{z_i \times 3}$, $B_{m_i \times z_i}$, $H_{m_i \times z_i}$
Output: t_{all}^{i}

Initialization: set the array I, the subtask queue $\boldsymbol{Q}$ and the set of predecessor nodes $\boldsymbol{R}$ to $\varnothing$
Use the constraint relationship G_i to set the array $I(i)$
Enqueue the ith subtask with $I(i) = 0$ to $\boldsymbol{Q}$ and set the number of traversed subtasks $u = 0$, the number of subtasks in the current layer k to the current queue size
while $\mathbf{Q}! = \varnothing$ **do**
  if $u = k$ **then**
    $u = 0, k = \text{size}(\mathbf{Q})$.
  end if
  The subtask is dequeued, and the task is expressed as v, $u+ = 1$
  for $\mathbf{i} \longleftarrow 0$ to $z_{\mathbf{i}}$ **do**
    if there exists a directed edge of **v** to ***i*** **then**
  Add the **v**th subtask and its predecessor node set $\mathbf{R(v)}$ to $\boldsymbol{R}\mathbf{(i)}$ $I(i)- = 1$
      if $I(i) = 0$ **then**
  enqueue the ith subtask to $\boldsymbol{Q}$
  end if
    end if
  end for
end while
According to $B_{m_i \times z_i}$, the subtasks are assigned to edge nodes.
Initialization: set the subtask completion list O to $\varnothing$, set the remaining execution latency of subtasks Y by $C_{m_i \times z_i}$, and set the current running time $h = 0$
while $O < z_i$ **do**
  Determine the subtask to be assigned to each edge node, which satisfies the direct predecessor set is subset of O
  Find the minimum execution latency w from the currently executed subtasks in parallel
  $Y(i)- = w$, when $Y(i) = 0$, add the ith subtask to O and set $h+ = w$
end while
return h

Algorithm 3: Algorithm implementation.

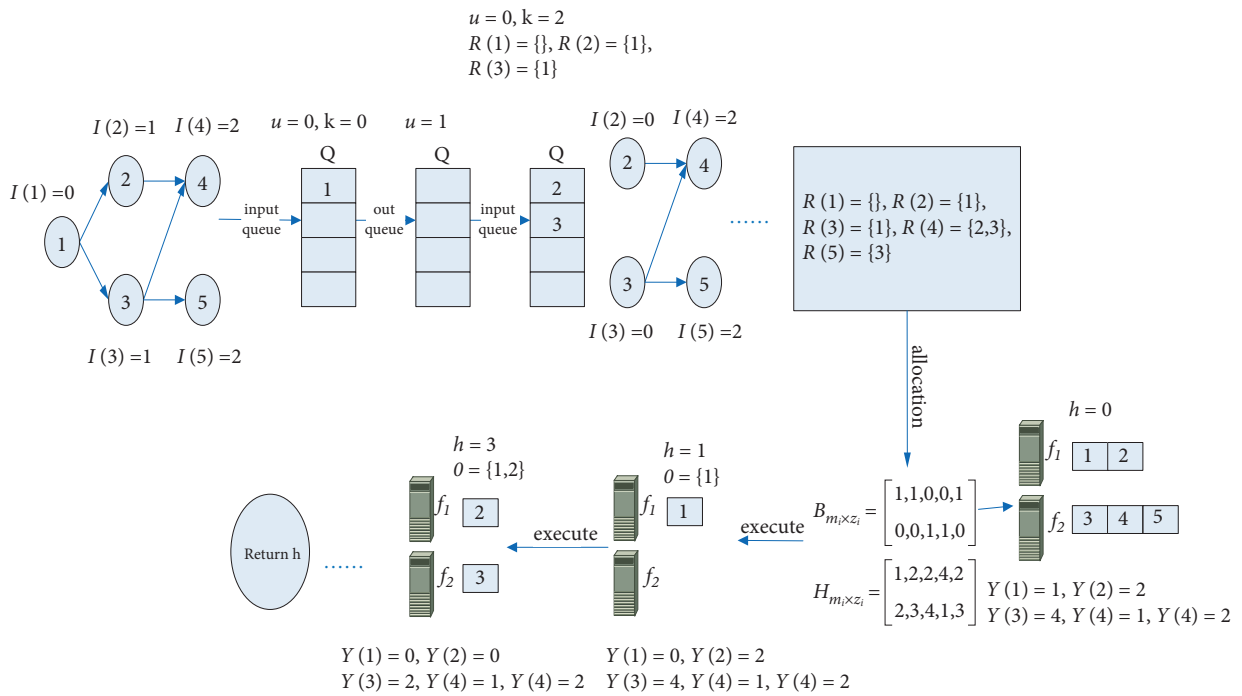

Figure 4: The calculation process of execution latency.

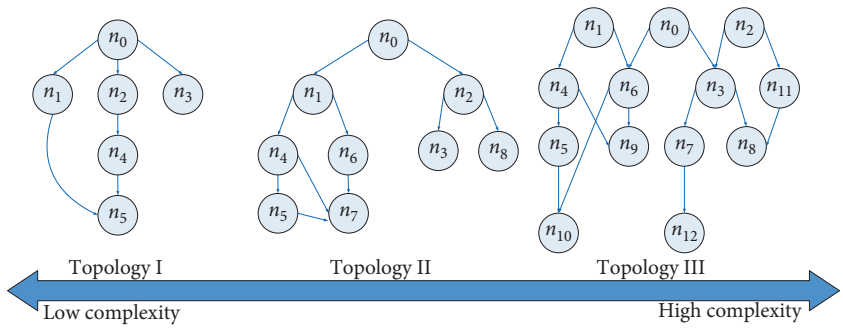

Figure 5: Topological structure of reasoning tasks.

Table 2: Constraint parameters of simulation experiment.

Parameter	Value
Task data volume	3–5Mbits
Task complexity	1200–3000 megacycles
Vehicle computing power	1-2GPOS
Edge node transmission rate	100–120Mbits/s
Edge node computing power	10–15GPOS
Number of scene edge nodes	1–4

Table 3: Completion latency (ms) of different reasoning tasks in various edge environments.

	Traditional rule			Priority rule		
	$n = 6$	$n = 9$	$n = 13$	$n = 6$	$n = 9$	$n = 13$
$m = 1$	6276	6891	7687	6276	6891	7687
$m = 2$	1025	1329	1849	1025	1329	1849
$m = 3$	840	862	1119	668	845	1078
$m = 4$	728	793	949	668	751	800

based on two rules: traditional rule and priority rule. Traditional rule executes the subtasks according to their corresponding topology depths [23] and priority rule executes the ones according to the priority evaluation for each subtask discussed in Section 4.1.

From Table 3, we find that the completion latency of reasoning tasks reduces as the number of edge nodes increases. Under the same circumstances, the priority rule for subtask execution can effectively reduce the completion latency of reasoning tasks compared to the traditional rule. As the topology complexity of reasoning tasks increases, this gap will become larger. This is because that the maximum number of parallel subtasks in the same time is limited by the number of edge nodes, and the priority rule can increase the upper limit number of the parallel subtasks.

Figure 6 shows the average completion latency of different scheduling algorithms (i.e., GA-PSO, Q-learning, and DQN) with different seasoning tasks in various edge environments, where m denotes the number of edge nodes in each experiment. In each subgraph, we record the completion latency of reasoning tasks with different topologies for 100 rounds and display the average completion latency of reasoning tasks for every 10 rounds. From Figure 6(a), we find that GA-PSO is difficult to converge, although it can get a feasible solution with shorter completion latency of reasoning tasks. In contrast, DQN can not only get a feasible solution with shorter completion latency, but also convergence well. From Figure 6(b) and Figure 6(c), the performance of Q-learning is similar to that of DQN when the numbers of subtasks and edge nodes are both small. However, the convergence performance of Q-learning will decrease as the topology complexity of reasoning tasks increases. The main reason for the different scheduling results with various algorithms is that the increase in the number of subtasks has brought about the multiplication of the number of solutions in searching space. For GA-PSO, finding the optimal solution mainly relies on randomness and fitness function. Therefore, when the number of feasible solutions in searching space is huge, GA-PSO is easy to fall into the local optimal solution. Q-learning is difficult to build the Q list and converge also due to the huge number of feasible solutions in searching space. However, DQN converts the Q list to the Q value function by neural network, which can solve the problem with a huge number of states (i.e., feasible solutions) and make it easier to converge.

Table 4 shows the average runtime (s) of different algorithms with different seasoning tasks in various edge environments. Each grid in Table 4 is the average of the runtime of 100 rounds for different algorithm. From Table 4, we find that the runtime of GA-PSO is relatively stable with different seasoning tasks in various edge environments. This is because that the runtime of GA-PSO mainly depends on

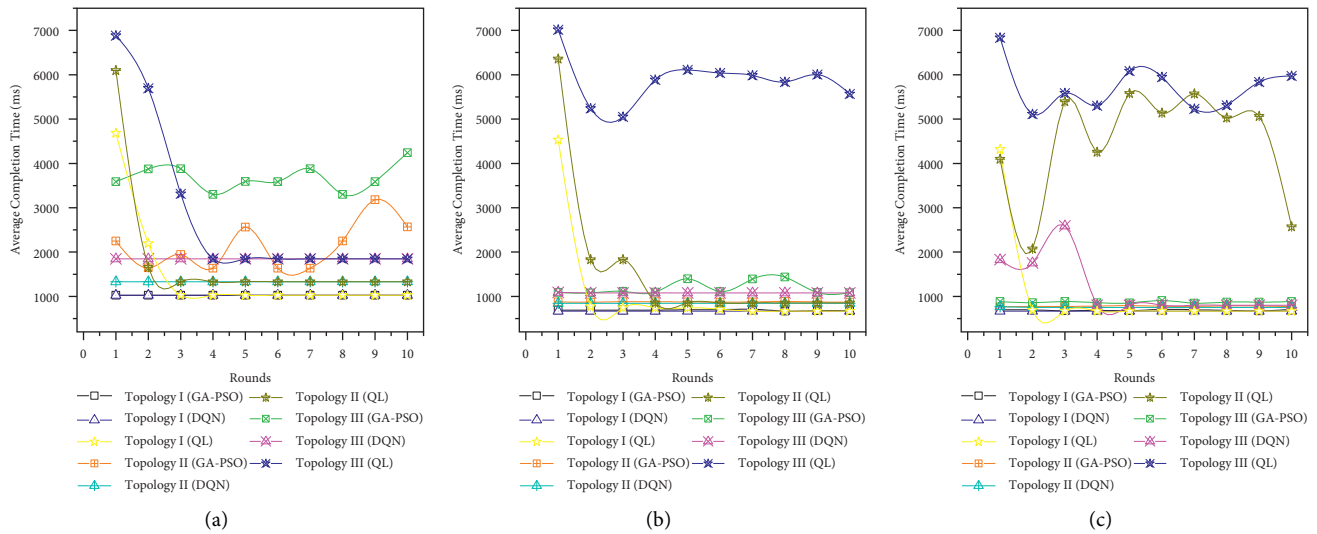

Figure 6: Average completion latency of different scheduling algorithms with different seasoning tasks in various edge environments. (a) $m = 2$. (b) $m = 3$. (c) $m = 4$.

Table 4: Average runtime (s) of different algorithms with different seasoning tasks in various edge environments

Number of edge nodes	Algorithm	Topology I ($n = 6$)	Topology II ($n = 9$)	Topology II ($n = 13$)
$m = 2$	GA-PSO	0.53	0.76	2.02
	DQN	**0.02**	**0.05**	**0.08**
	Q-learning	**0.02**	0.06	0.44
$m = 3$	GA-PSO	0.58	0.72	1.19
	DQN	**0.04**	**0.12**	**0.28**
	Q-learning	0.10	0.58	11.36
$m = 4$	GA-PSO	0.61	0.83	0.93
	DQN	**0.07**	**0.13**	**0.92**
	Q-learning	0.19	7.77	11.29

the number of particles used in the update process, which is relatively stable even if the edge environments change during the scheduling process. The average runtime of DQN and Q-learning is better than that of GA-PSO, and DQN performs best with different seasoning tasks in various edge environments. This is because the runtime of RL algorithms will decrease as the number of feasible solutions learned increases. In addition, the architecture of the neural network used in DQN is more suitable for reasoning tasks scheduling in vehicle edge computing, compared with the Q list used in Q-learning.

6. Conclusions

This paper proposes a scheduling strategy based on DQN for reasoning tasks in vehicle edge computing, which aims to reduce the completion latency of reasoning tasks. The extensive simulation experiments show that the proposed strategy can achieve the superior performance compared to other classic methods. Our strategy and other classic methods all perform well when the structure of reasoning tasks is simple, while GA-PSO has poor convergence. Specially, our strategy has better performance and convergence than any other classic methods when the structure of reasoning tasks is complex.

In the future, we will improve the scheduling algorithm through optimizing the training efficiency of the neural network to fit the wireless channel fluctuations and radio interference in vehicle edge computing. In addition, we will further consider a multivehicle collaborative scheduling strategy to alleviate uneven resources allocation for multivehicle tasks in edge environments.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

This work was presented in part at the 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications (ISPA) with the title "A Time-Driven Workflow Scheduling Strategy for Reasoning Tasks of Autonomous Driving in Edge Environment."

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partly supported by the Intelligent Computing and Application Research Team of Concord University College, Fujian Normal University under Grant no. 2020TD001 Natural Science Foundation of China under Grant no. 62072108; Project on the Integration of Industry and Education of Fujian Province under Grant no. 2021H6026 Natural Science Foundation of Fujian Province under Grant nos. 2019J01286 and 2019J01427; and Young and Middle-Aged Teacher Education Foundation of Fujian Province under Grant no. JT180098.

References

[1] J. Cheng, G. Yuan, M. Zhou et al., "Accessibility analysis and modeling for IoV in an urban scene," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4246–4256, 2020.

[2] J. Wang, Y. Qiao, and H. Hong, "Graph-based auto-driving reasoning task scheduling," *Journal of Computer Research and Development*, vol. 8, pp. 68–77, 2017.

[3] Z. Gao, T. Sun, and L. He, "Causal reasoning decision-making for vehicle longitudinal automatic driving," *Journal of Jilin University(Engineering and Technology Edition)*, vol. 49, no. 5, pp. 1392–1404, 2019.

[4] R. Xie, X. Lian, and Q. Jia, "Survey on computation offloading in mobile edge computing," *Journal on Communications*, vol. 39, no. 11, pp. 138–155, 2018.

[5] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, "Efficient task completion for parallel offloading in vehicular fog computing," *China Communications*, vol. 16, no. 11, pp. 42–55, 2019.

[6] Z. Xiao, F. Li, H. Jiang et al., "A joint information and energy cooperation framework for CR-enabled macro-femto heterogeneous networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2828–2839, 2020.

[7] Z. Xiao, X. Dai, H. Jiang et al., "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2038–2052, 2020.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[9] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[10] J. Fang, Z. Zhang, and X. Zhang, "Cloud workflow scheduling algorithm based ON trade-off fitness," *Computer Applications and Software*, vol. 36, no. 5, pp. 255–261, 2019.

[11] A. Ghorbannia Delavar and Y. Aryan, "HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems," *Cluster Computing*, vol. 17, no. 1, pp. 129–137, 2014.

[12] Y. Xie, Y. Zhu, Y. Wang et al., "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud-edge environment," *Future Generation Computer Systems*, vol. 97, pp. 361–378, 2019.

[13] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[14] T. Q. Thinh, J. Tang, and Q. D. La, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[15] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[17] X. Xiong, K. Zheng, and L. Lei, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 99, p. 1, 2020.

[18] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.

[19] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432–7445, 2017.

[20] T. Zhao, T. Zhang, and Y. Chen, "Task distribution offloading algorithm of vehicle edge network based on DQN," *Journal on Communications*, vol. 41, no. 10, pp. 172–178, 2020, in chi.

[21] G. Rjoub, J. Bentahar, and O. Abdel Wahab, "Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, 2020.

[22] Q. Qi, L. Zhang, J. Wang et al., "Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13861–13874, 2020.

[23] K. Lin, B. Lin, and X. Chen, "A time-driven workflow scheduling strategy for reasoning tasks of autonomous driving in edge environment," in *Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications*, ISPA, Shenzhen, China, December 2019.

[24] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[25] M. Chen, U. Challita, and W. Saad, "Artificial neural networks-based machine learning for wireless networks: a tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2017.

[26] G. Rjoub, J. Bentahar, and O. A. Wahab, "BigTrustScheduling: trust-aware big data task scheduling approach in cloud computing environments," *Future Generation Computer Systems*, vol. 110, pp. 1079–1097, 2020.

[27] H. Ghunaim and J. Dichter, "Applying the FAHP to improve the performance evaluation reliability of software defect classifiers," *IEEE Access*, vol. 7, pp. 62794–62804, 2019.

[28] Z. Geng, Z. Wang, C. Peng, and Y. Han, "A new fuzzy process capability estimation method based on kernel function and FAHP," *IEEE Transactions on Engineering Management*, vol. 63, no. 2, pp. 177–188, 2016.

[29] Y. Lu, B. Lin, and M. Liu, "The performance evaluation of university scientific research Project management based on

the FAHP," *Journal of Digital Information Management*, vol. 12, no. 1, pp. 18–25, 2014.

[30] F. Xiao, "EFMCDM: evidential fuzzy multicriteria decision making based on belief entropy," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1477–1491, 2020.

[31] T. Li and L. She, "Retail location decision-making based on the combination of AHP and entropy weight," in *Proceedings of the 2010 Third International Joint Conference on Computational Science and Optimization*, Huangshan, China, May 2010.

[32] B. Lin, F. Zhu, J. Zhang et al., "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4254–4265, 2019.

[33] Y. Liu, L. Liu, and L. Zheng, "Study on the downlink performance of roadside unit in vehicular ad-hoc networks," *Journal of Software*, vol. 26, no. 7, pp. 1700–1710, 2015.