Reliability-Enhanced Task Offloading Strategy for Intelligent Connected Vehicles in Fog Computing

Wufei Wu, *Senior Member, IEEE*, Jia Wei, Xiaochuan Guo, Jiaxin Zeng, Gang Zeng, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Fog computing architecture has the characteristics of high timeliness and low latency due to its proximity to data sources. This brings new opportunities for the rapid development of intelligent connected vehicle (ICV) technology. However, the existing research on fog computing task offloading methods lacks consideration of the functional safety of ICVs. Therefore, this paper introduces the hardware failure rate and communication interruption rate of on-board equipment into fog computing task offloading to solve the problem of ICV task offloading under energy and priority constraints. Furthermore, we design a reliability-enhanced generative adversarial imitation learning algorithm (REGAIL) for energy-constrained task offloading. The algorithm can be divided into two parts: pre-scheduling and energy allocation. First, tasks are pre-allocated by the shortest completion time; second, the generative adversarial imitation learning algorithm (GAIL) and the proximal policy optimization algorithm (PPO) are combined for policy training; finally, we evaluate the performance of these algorithms using a randomly generated directed acyclic graph (DAG) as an application scenario. Experimental results show that compared with existing algorithms, our algorithm not only achieves stable training, but also improves execution time by 5.56% and reliability by 20.19% .**

***Index Terms*—Application reliability, directed acyclic graphs (DAG), energy constraints, fog computing, generative adversarial imitation learning (GAIL), intelligent and connected vehicle (ICV).**

I. Introduction

A. Background

With the continuous development of intelligent and connected vehicle technology (ICV), vehicles are facing increasingly complex data interaction and processing requirements. In practical applications, vehicles need to process a large amount of real-time data, including traffic information, vehicle status, and navigation data. This puts tremendous pressure on the computational resources of in-vehicle systems [1]. Particularly with the rise of electric vehicles (e.g., Tesla) in the market, running applications with high complexity to meet users' Quality of Experience (QoE) expectations becomes challenging due to limited battery capacity and computational resources. Although traditional cloud computing can alleviate the resource constraints of vehicles, due to the cloud's location in the network core, vehicles are often situated far from it. Therefore, long-distance task offloading leads to significant communication latency, which fails to meet the low-latency service requirements of vehicles [2], [3]. Moreover, during the task offloading process, issues such as data transmission interruptions, in-vehicle device failures, and task execution failures may occur due to factors such as in-vehicle devices and network environments. These issues can negatively impact the stability and reliability of applications. Among them, reliability is the frequency of successful execution of plans, which has been widely regarded as an increasingly relevant issue in service-oriented computing systems [4], [5]. Therefore, designing reliable task offloading strategies for vehicles to maximize the reliability of task offloading has become an important research direction in the field of ICV.

B. Challenges and Motivation

Fig. 1 shows some commonly used network computing paradigms in task offloading for current IoT applications. Among them, the practical value of fog computing in this field has been continuously highlighted and has been extensively studied [6], [7], [8], it can offload the computing tasks of vehicle-mounted applications to mobile edge cloud (MEC) servers. This not only effectively enhances the service level and computing capacity at the edge, but also brings computing closer to the network edge, significantly reducing communication latency and improving user experience, thus providing an excellent solution to the aforementioned issues. For the sake of clarity, we will use MEC to refer to mobile edge cloud servers in the following text.

MEC revolutionizes on-board vehicular capabilities by offering localized computational and storage resources at the network edge. This proximity to vehicles significantly improves server responsiveness, thereby enhancing the users' experience. Within the MEC framework, edge servers serve as complementary

Received 1 May 2025; revised 15 August 2025; accepted 3 November 2025. Date of publication 10 November 2025; date of current version 18 May 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62462045, in part by Jiangxi Province Science and Technology Development Program under Grant 20232BAB202009 and Grant 320244BBC30006, and in part by China Postdoctoral Science Foundation under Grant 2020TQ0134. The review of this article was coordinated by Mr. Ajeya Gupta. *(Corresponding authors: Wufei Wu; Xiaochuan Guo.)*

Wufei Wu, Jia Wei, and Jiaxin Zeng are with the School of Information Engineering, Nanchang University, Nanchang 330000, China (e-mail: wuwufei@ncu.edu.cn; weijia@email.ncu.edu.cn; JiaxinZeng@email.ncu.edu.cn).

Xiaochuan Guo is with the School of Information Science and Engineering, Hunan University, Hunan 410000, China (e-mail: lumanchuan@email.ncu.edu.cn).

Gang Zeng is with the Graduate School of Engineering, Nagoya University, Nagoya 464-8603, Japan (e-mail: zeng.gang.s6@f.mail.nagoya-u.ac.jp).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TVT.2025.3631407

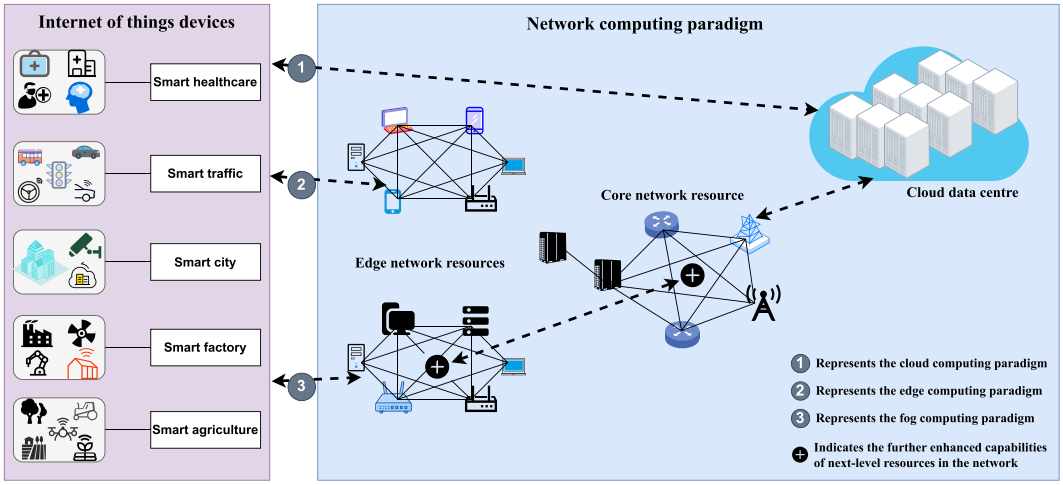

Fig. 1. Schematic of IoT applications and environments with computing paradigm support [9].

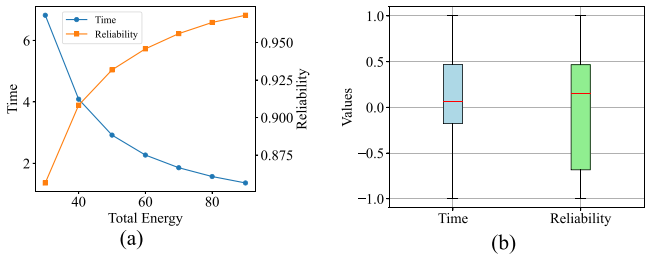

Fig. 2. Comparison of execution time and energy. (a) Line graph of application execution time and reliability under different energy levels using the average energy allocation method. (b) Box plot of execution time and reliability for 20 instances under the same energy level using the random energy allocation method.

processing power, enabling vehicles to offload computationally intensive tasks through a process known as task offloading [8]. MEC serves a dual function in enhancing vehicle technology. By increasing in-vehicle resources, it compensates for device limitations and enhances application performance, while also extending battery life [10]. Simultaneously, MEC acts as a partial substitute for remote cloud servers, reducing response latency for vehicle applications [2]. In other words, MEC not only improves the functionality and efficiency of in-vehicle devices but also provides faster and more reliable connectivity for vehicle applications.

However, due to energy constraints, how to allocate energy to each task in MEC and fog computing systems faces certain challenges. The choice of energy allocation strategy has a crucial impact on task offloading performance and system reliability. In response to this, this study designs two experiments for verification, and the experimental results are shown in Fig. 2. Among them, Fig. 2(a) illustrates a line graph showing the execution time and reliability of the same application under different total energy levels using the average energy allocation method (where each task receives the same amount of energy). We observe that the rate of decrease in execution time and increase in reliability are nearly identical. This suggests that with the average energy allocation method, the impact of total energy on both execution time and reliability is consistent. Fig. 2(b) presents a box plot showing the reliability of execution time over 20 instances using random energy allocation methods. The reliability variance is 0.4556, while the execution time variance is 0.2871, indicating that the execution time is relatively stable compared to reliability. It can be seen from Fig. 2 that different energy allocation strategies lead to different results, so designing an efficient energy allocation strategy is of great significance. In response to this phenomenon, this paper proposes an energy allocation method for task offloading, aiming to enhance task offloading and reliability while minimizing the unnecessary increase in execution time.

C. Our Contributions

In this study, our primary objective is to address the reliability-enhanced task offloading problem under energy and priority constraint in a fog computing environment. The contributions of this study are as follows.

- We propose a method that utilizes the minimum fading coefficient to estimate the transmission time of wireless channels, which can meet the energy constraints of the application.
- We propose a more precise concept of reliability and establish a reliability model in fog computing. This model is grounded in the device failure rate, modeled using a Poisson distribution, and the communication interruption rate, formulated with a Rayleigh fading channel model.
- We develop an energy allocation algorithm to improve the efficiency and reliability of task offloading, which combines generative adversarial imitation learning (GAIL) with proximal policy optimization (PPO). Furthermore, we evaluate the proposed strategy through a large number of experiments and verify the effectiveness of this algorithm.

II. Related Work

Currently, a large number of studies have been concentrated on task offloading in fog computing, some of the work is shown in Table I. Of these, most of the work focuses on optimizing

TABLE I
RESEARCH ON TASK OFFLOADING IN FOG COMPUTING

Work	Task Properties		Architectural Properties		Optimization Objective			Reliability Indicator
	Task Delay Constraint	Task Dependency	Multiple MECs	System Model	Time	Energy	Reliability	
[11]	✓	×	✓	Edge&Cloud	✓	✓	×	-
[12]	×	×	✓	Edge&Cloud	✓	✓	×	-
[13]	✓	×	✓	Edge	✓	✓	✓	Single
[14]	×	×	✓	Fog&Cloud	✓	×	✓	Single
[15]	×	✓	✓	Fog	✓	constraint	✓	Single
[16]	×	✓	✓	Cloud	constraint	✓	constraint	Single
[17]	×	✓	✓	Edge	✓	×	✓	Single
[18]	✓	✓	✓	Edge	constraint	constraint	✓	Single
[19]	×	×	✓	Edge	✓	✓	×	-
[20]	×	×	✓	Edge	constraint	✓	×	-
Our scheme	×	✓	✓	Fog	✓	constraint	✓	Multiple

energy consumption and latency, but lacks the guarantee of task offloading reliability. For example, Liu et al. [11] considered the characteristics of tasks and mobile edge computing, and classified tasks into indivisible and divisible tasks based on data size (i.e., whether partitioning affects functionality). Based on this, they proposed binary offloading algorithms and partial offloading algorithms, effectively reducing latency and energy consumption. In [12], Sharma et al. combined first-order meta-learning and deep Q learning to solve the problem of time and energy consumption in multi-task offloading. As noted above, previous studies have focused on energy consumption and latency, often neglected the reliability. However, the reliability of task offloading is of vital importance. In real scenarios, tasks offloaded from vehicle applications to MEC for execution may be interrupted due to factors such as device and communication failures. Once the offloading process fails, it may cause decision delays, and even lead to the spread of local failures, resulting in safety accidents.

For the reliability of task offloading, many studies have proposed different evaluation methods. Fan et al. [13] regard the inference error rate as part of the task processing cost, which is incorporated into the calculation of the task processing cost through a weight factor, indirectly considering the reliability of task processing. Additionally, some studies rely on statistical probability methods to model system reliability. Fayez et al. [14] determined the failure rate of fog nodes by analyzing fault logs, enabling reliability-sensitive tasks to avoid nodes with high failure rates. Liu et al. [15] used the Poisson distribution to estimate the failure rate of communication links and designed an algorithm to optimize the reliability of task offloading. In addition, since the communication between vehicles and MEC usually relies on wireless networks and is affected by channel fading, communication reliability has become a crucial element in task offloading. A growing number of studies use the communication outage probability to evaluate channel reliability. A famous study [21] analyzed the theoretical reliability boundary of wireless fading channels using Rayleigh fading and evaluated the communication interruption probability as a performance indicator. In [16], Azimi et al. used a similar channel and reliability model to study the trade-off between energy consumption and delay in mobile cloud computing applications under fading channel conditions. Building on prior reliability studies, we integrate a hardware failure rate modeled by Poisson distribution and a communication interruption rate modeled using a Rayleigh fading channel as reliability metrics. Compared with a single indicator that only reflects one aspect of reliability, we combine two indicators to consider reliability, which can more accurately describe the actual performance of the system. The hardware failure rate can reflect the stability of the task processing carrier, and the communication failure rate can reflect the stability of data transmission.

In addressing the reliability optimization problem in task offloading, traditional algorithms generally suffer from poor real-time performance and adaptability. Liu et al. [17] studied the trade-off between latency and reliability when offloading tasks to MEC in ultra-reliable low-latency communication (URLLC) scenarios, and proposed a corresponding non-convex optimization problem framework and various optimization methods, such as algorithms based on reformulation-linearization technique. Shang et al. [18] investigated the reliability optimization problem under latency and energy constraints in mobile edge computing environments, and introduced the maximum reliability optimization algorithm (MROA), which uses the traditional list scheduling.

In real-world road scenarios, a vehicle typically has multiple options for MEC [22], [23]. In recent years, task scheduling issues in multi-MEC systems have garnered widespread attention. In [19], Tran et al. investigated the joint problem of task offloading and resource allocation in multi-cell, multi-server MEC systems. They optimized task offloading decisions, user uplink communication power settings, and MEC computational resource allocation to maximize the benefits of task offloading, including reducing user latency and energy consumption. Bi et al. [20] investigated the computation offloading and resource allocation problem in heterogeneous multi-MEC environments. Task scheduling strategies become more complex and challenging in multi-MEC environments. To apply the theory to practice, this study also investigates a fog computing environment consisting of one vehicle and multiple MECs.

Generally speaking, most current studies aim to reduce latency and energy consumption in task offloading, while few studies consider the reliability issue in this process. In addition, they usually use a single indicator to evaluate reliability, which can only reflect one aspect of reliability. At the same time, in real-world intelligent connected vehicle offloading scenarios, multiple MECs are often involved. Therefore, this study

TABLE II
MAIN NOTATIONS AND DEFINITIONS

Notations	Definitions
$\mathcal{A}$	An application for vehicle
M	Set of heterogeneous servers
$\widetilde{E}$	The total energy constraint value of $\mathcal{A}$
L	The task list for $\mathcal{A}$
r_i	The execution requirement of task t_i
d_i	The communication requirement of task t_i
w_k	The channel bandwidth between vehicle and MEC$_k$
β_k	A constant that integrates factors including the background noise, the neighbor channel interference, and the channel gain between vehicle and the MEC$_k$
$h_{i,k}$	The power attenuation coefficient of t_i from vehicle to MEC$_k$
$h_{k,min}$	Minimum power attenuation coefficient
$c_{i,k}$	The communication speed of t_i from UE to MEC$_k$
s_k	The execution speed of MEC$_k$
$s_{0,i}$	The execution speed of t_i in vehicle
λ_k	The constant failure rate of MEC$_k$ per unit time
T_i^{comp}	The time it takes for t_i to perform calculations
T_i^{comm}	Time spent on communication in t_i
R_i^{comp}	Computation reliability of t_i
R_i^{comm}	Computation reliability of t_i
$E_{\min}(\mathcal{A})$	Minimum energy required for application execution
$\widetilde{E}$	Energy constraints applied
E_i	The total energy consumed for t_i

considers a multi-MEC environment and focuses on enhancing the reliability of task offloading for vehicle applications in fog computing environments that incorporate multiple indicators. Furthermore, considering that energy is a scarce resource for ICVs [24], [25], [26], particularly electric vehicles, we establish a threshold as the upper bound for ICV energy consumption. This study designs an efficient optimization strategy by combining GAIL and reinforcement learning algorithms, which can meet energy constraints, reduce system delay, improve system reliability, and ensure real-time application performance. In the following sections, we will address the enhancement of task offloading reliability for vehicle applications in fog computing environments. We aim to provide new design references for system designers and developers in this field.

III. System Model

In this section, we present task offloading models for vehicle applications. We study a heterogeneous fog computing environment comprising a single vehicle and n MECs, as illustrated in Fig. 3. Within this environment, the vehicle initiates a service request in the form of a mobile application, which is subsequently stored in the mobile application pool. The task scheduler extracts applications from this pool and decomposes them into tasks with priority constraints. These tasks can be executed either locally on the vehicle or offloaded to the MECs for processing. We need to ensure that the energy consumed to execute the application does not exceed the energy constraint $\widetilde{E}$. The system encompasses $n+1$ servers, collectively represented as $M = \{\text{MEC}_0, \text{MEC}_1, \text{MEC}_2, \ldots, \text{MEC}_n\}$. Here, MEC$_0$ denotes the vehicle itself, while $M' = \{\text{MEC}_1, \text{MEC}_2, \ldots, \text{MEC}_n\}$ signifies $n+1$ heterogeneous fog computing servers.

A. The Application Model

This study investigates the problem of task offloading for vehicle applications. Assuming a vehicle with an application denoted

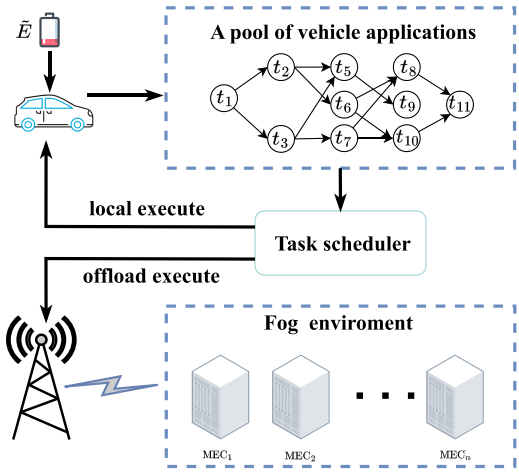

Fig. 3. Illustration of service-oriented fog computing environment.

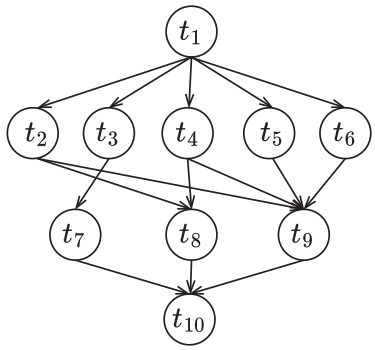

Fig. 4. Schematic diagram of DAG type application.

as $\mathcal{A} = (L, \prec)$, the application comprises a task list represented by $L = (t_1, t_2, \ldots, t_m)$. Each task $t_i = (r_i, d_i)$ is characterized by its computational demand r_i in billions of processor cycles or instructions and its communication requirement d_i in megabits (MB) for data transfer between the vehicle and MEC.

Vehicle applications with task priority constraints can be represented using directed acyclic graphs (DAGs) [6], [8], [18], [27], as shown in Fig. 4. In this graph, each node represents a task within the application, and directed edges indicate the task priority relationships. Additionally, we use $t_i \prec t_j$ to indicate that t_i has higher priority than t_j. The sets of immediate predecessor and successor tasks of t_i are denoted by $pred(t_i)$ and $succ(t_i)$, respectively. Tasks can only be executed when all their predecessors are completed. A task without predecessors in the graph is defined as the entry task of application $\mathcal{A}$, denoted as t_{entry}, while a task without successors is referred to as the exit task of the application, denoted as t_{exit}.

B. The Communication Model

The wireless communication environment between vehicles and MEC is typically regarded as a block fading environment. Therefore, we assume that the channel fading coefficient between the vehicle and MEC remains constant during mission transmission [1]. Assume that $P_{i,k}$ is the transmission energy consumption when the task t_i is offloaded from the vehicle to

the MEC_k, which is determined by the vehicle, and accordingly, the actual communication speed $c_{i,k}$ is

$$c_{i,k} = w_k \log_2(1 + P_{i,k}\beta_k |h_{i,k}|^2), \tag{1}$$

where w_k depicts the channel bandwidth between vehicle and the MEC_k. $\beta_k = ID_k^{-a}/N_0$ [28], where I is the system constant, N_0 is the noise power, D_k is the distance between the vehicle and the MEC_k, and a is the path loss exponent. $h_{i,k}$ is a random coefficient used to characterize the fading of the wireless channel, which considers factors of multipath effects and shadow effects, etc. We also consider the Rayleigh Fading, whose probability density obeys the Rayleigh Distribution, and the corresponding power attenuation coefficient $|h_{i,k}|^2$ follows an exponential distribution with mean σ_k^2 [1], [29], i.e.:

$$f_{k^2}(x) = \frac{1}{\sigma_k^2} e^{\frac{x}{\sigma_k^2}}. \tag{2}$$

If the vehicle offloads t_i to the MEC_k for remote execution, the corresponding communication time (in Seconds) is $T_{i,k}^{comm} = d_i/c_{i,k}$, and the communication energy consumption of vehicle (in Joules) is

$$\begin{aligned} E_{i,k}^{comm} &= P_{i,k} T_i^{comm} \\ &= (2^{c_{i,k}/w_i} - 1) d_i / (\beta_k |h_{i,k}|^2 c_{i,k}). \end{aligned} \tag{3}$$

Given that the amount of output data after task execution is typically small, the time and energy consumption of the backhaul link communication is neglected here [17]. The randomness of the channel fading coefficients prevents the vehicle from accurately predicting the communication speed and time for t_i. This information can only be obtained after the communication is completed. According to [30], for (3), $E_{i,k}^{comm}$ is a monotonically increasing function of $c_{i,k}$, and when $c_{i,k} \in (0, \infty)$, there is $E_{i,k}^{comm} \in (\frac{d_i \ln 2}{w_k \beta_k |h_{i,k}|^2}, \infty)$, for all $1 \le k \le n$.

If the task t_i is executed locally at vehicle, then communication time $T_{i,0}^{comm} = 0$, and the vehicle communication energy consumption $E_{i,0}^{comm} = 0$.

C. The Computation Model

Assume that there are n heterogeneous fog nodes $M' = \{\text{MEC}_1, \text{MEC}_2, \ldots, \text{MEC}_n\}$, for each node MEC_k $(1 \le k \le n)$, let s_k denote its computation speed—measured by GHz or billion instructions per second—which is a fixed parameter independent of vehicle control. For MEC_0, the variable $s_{0,i}$ represents the computational speed and can be adjusted according to the actual needs.

If t_i is offloaded and executed remotely on MEC_k by the vehicle, the computation time (measured by seconds) of t_i is $T_{i,k}^{comp} = r_i/s_k$. In our model, we concentrate on the vehicle and study the offloading of the tasks, so the energy consumption required for the calculation is not considered, i.e., the corresponding calculated energy consumption (in joules) at this situation is $E_{i,k}^{comp} = 0$, for all $1 \le i \le m, 1 \le k \le n$.

If t_i is not offloaded and executed locally on the vehicle with computation speed $s_{0,i}$, then the computation time of t_i is $T_{i,0}^{comp} = r_i/s_{0,i}$. As to the vehicle, the power consumption (in Watts) consists of two main components, namely static power consumption and dynamic power consumption P_d. The former is often a constant, while for the latter we have $P_d = \xi s_0^\alpha$ as its definition, where s_0 is the computation speed of the vehicle and both ξ and α are constants related to the actual technologies, thus making the vehicle computation energy consumption as $P = P_d + P_s$. Therefore, the vehicle computation energy consumption corresponding to t_i in such a situation is $P_i = \xi s_{0,i}^\alpha + P_s$, and the vehicle computation energy consumption is

$$E_{i,0}^{comp} = P_i T_{i,0}^{comp} = ((\xi s_{0,i}^\alpha + P_s)/s_{0,i}) r_i, \tag{4}$$

for all $1 \le i \le m$. According to [30], it is evident that for (4), $E_{i,0}^{comp}$ is a monotonically increasing function of $s_{0,i}$, and when $s_{0,i} \in [(\frac{P_s}{\xi(\alpha-1)})^{1/\alpha}, \infty)$, there is $E_{i,k}^{comp} \in [r_i P_s^{1-1/\alpha} \xi^{1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}, \infty)$.

D. Execution Time and Energy Consumption

As for task t_i, the total execution time includes both the computation and the communication time, i.e., $T_i = T_i^{comm} + T_i^{comp}$. If t_i is executed locally on the vehicle, then we have $T_{i,0} = T_{i,0}^{comp} = r_i/s_{0,i}$. If t_i is executed remotely on MEC_k by the vehicle, then we have $T_{i,k} = T_{i,k}^{comp} + T_{i,k}^{comm} = r_i/s_k + d_i/c_{i,k}$, for all $1 \le i \le m, 1 \le k \le n$.

As for energy consumption, this study focuses on vehicle-oriented offload optimization, so the energy consumption at the MEC side is not taken into consideration, thus the execution energy consumption of the task t_i only contains two parts: the computation and communication energy consumption of the vehicle, i.e., $E_i = E_i^{comm} + E_i^{comp}$. If t_i is executed locally on the vehicle, then we have $E_i = E_{i,0}^{comp} = ((\xi s_{0,i}^\alpha + P_s)/s_{0i}) r_i$. If t_i is executed remotely on MEC_k by the vehicle, then we have $E_i = E_{i,k}^{comm} = P_{i,k} T_{i,k}^{comm}$, for all $1 \le i \le m, 1 \le k \le n$. According to the energy consumption mentioned above, the total energy consumption of the vehicle application $\mathcal{A}$ is $E(\mathcal{A}) = \sum_{i=1}^m E_i$.

Based on the analysis of task execution energy consumption in Sections III-B and III-C, there exists a theoretical lower bound $E_{i,k}^*$ for the execution energy consumption of t_i. Whether t_i is executed locally at the vehicle or offloaded to the MEC for execution, the minimum value of this metric represents the minimum energy required to ensure the executability of task t_i in the current heterogeneous fog environment. Therefore, the minimum scheduling energy of t_i is defined as follows

$$E_{i,k}^* = \begin{cases} r_i P_s^{1-1/\alpha} \xi^{1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}, & k = 0 \\ d_i \ln 2 / w_k \beta_k |h_{i,k}|^2, & 1 \le k \le n, \end{cases} \tag{5}$$

$$E_{i,\min} = \min_{0 \le k \le n} E_{i,k}^*. \tag{6}$$

The application allocates minimum energy to each task before scheduling tasks. Correspondingly, we can give the minimum schedulable energy of the vehicle application $\mathcal{A}$ as $E_{\min}(\mathcal{A}) = \sum_{i=1}^m E_{i,\min}$. Obviously, if the total energy consumption constraint for application $\mathcal{A}$ is $\widetilde{E}$, then $\mathcal{A}$ is schedulable only if $\widetilde{E}$ is greater than $E_{\min}(\mathcal{A})$.

For t_i, the available energy consists of two parts: the first part is the minimum energy $E_{i,\min}$, which ensures that the task can be executed; the second part is the energy allocated to the application ΔE_i. Thus, the energy $\widetilde{E_i}$ is expressed as $\widetilde{E_i} = E_{i,\min} + \Delta E_i$.

In doing so, we transform the application-level energy constraints into task-specific energy constraints for individual tasks. It is evident that satisfying the energy constraints of individual tasks is sufficient to ensure that the entire application meets the energy constraints.

If the energy constraint $\widetilde{E_i}$ for each task t_i is given, the corresponding computation speed $s_{0,i}$ can be calculated [6]. However, the channel fading coefficient $h_{i,k}$ and the actual communication speed $c_{i,k}$ are unpredictable to the vehicle so that the vehicle is unable to obtain the communication power $P_{i,k}$ for the task t_i according to (1) until the actual offloading of t_i is completed. The reference channel fading coefficient $h_{i,k}^{ref}$ is employed to determine the communication power $P_{i,k}$, so we have

$$\widetilde{E_i} = \frac{(2^{c_{i,k}^{ref}/w_k} - 1)}{\beta_k \left| h_{i,k}^{ref} \right|^2} \frac{d_i}{c_{i,k}^{ref}}, \tag{7}$$

and

$$P_{i,k} = \frac{(2^{c_{i,k}^{ref}/w_k} - 1)}{\beta_k \left| h_{i,k}^{ref} \right|^2}. \tag{8}$$

Assuming that during the scheduling of the vehicle application $\mathcal{A}$ there exists a minimum value $h_{k,min}$ of the channel fading coefficient between the vehicle and MEC$_k$ and it is available for the vehicle, which leads to $h_{i,k}^{ref} = h_{k,min}$, the proof is as follows.

Theorem III.1: During the scheduling process of vehicle application $\mathcal{A}$, the energy consumption E_i between vehicle and MEC$_k$ satisfies the condition $E_i \le \widetilde{E_i}$ only if it satisfies $h_{i,k}^{ref} = h_{k,min}$.

Proof: We use a counter-argument. Assume that we select $h_{i,k}^{ref} > h_{k,\min}$, there exists the case of the actual channel fading coefficient $h_{i,k} < h_{i,k}^{ref}$, at which point it is clear that the vehicle communication power is

$$P_{i,k} = \frac{(2^{c_{i,k}^{ref}/w_k} - 1)}{\beta_k \left| h_{i,k}^{ref} \right|^2} = \frac{(2^{c_{i,k}/w_k} - 1)}{\beta_k \left| h_{i,k} \right|^2}. \tag{9}$$

According to (9), the actual communication speed at this point satisfies the case $c_{i,k} < c_{i,k}^{ref}$, the actual communication energy used by the vehicle for offloading t_i will be

$$E_i^{comm} = P_{i,k} T_i^{comm} \frac{(2^{c_{i,k}^{ref}/w_k} - 1)}{\beta_k \left| h_{i,k}^{ref} \right|^2} \frac{d_i}{c_{i,k}} > E_i. \tag{10}$$

That is, the vehicle communication energy consumption can be guaranteed to satisfy the energy constraint only when $h_{i,k}^{ref} = h_{k,min}$ is taken. Obviously, there is an energy consumption balance for task offloading in such energy assignment method, i.e., $E_i \le \widetilde{E_i}$. ■

Based on Theorem III.1, we can modify (5) to

$$E_{i,k}^{*} = \begin{cases} r_i P_s^{1-1/\alpha} \xi^{1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}, & k = 0 \\ d_i \ln 2 / w_k \beta_k \left| h_{k,min} \right|^2, & 1 \le k \le n. \end{cases} \tag{11}$$

That is, t_i can be offloaded to MEC$_k$ if $\widetilde{E}_i > d_i \ln 2 / w_k \beta_k |h_{k,\min}|^2$. The proof is as follows.

Theorem III.2: t_i can be unloaded to MEC$_k$ only if $\widetilde{E}_i > d_i \ln 2 / w_k \beta_k |h_{k,\min}|^2$.

Proof: Considered using the worst-case channel fading scenario, which can be obtained when the energy allocated to t_i is $\widetilde{E_i}$ and offloaded to the MEC$_k$:

$$\widetilde{E_i} = \frac{(2^{c_{i,k}^{ref}/w_k} - 1)}{\beta_k \left| h_{k,\min} \right|^2} \frac{d_i}{c_{i,k}^{ref}}, \tag{12}$$

that is

$$2^{c_{i,k}^{ref}/w_k} - (\widetilde{E_i}/d_i) \beta_k c_{i,k}^{ref} \left| h_{k,\min} \right|^2 - 1 = 0. \tag{13}$$

For the $2^{c_{i,k}^{ref}/w_k}$ Taylor expansion we have

$$2^{c_{i,k}^{ref}/w_k} > 1 + (\ln 2/w_k) c_{i,k}^{ref} + \frac{1}{2} (\ln 2/w_k)^2 (c_{i,k}^{ref})^2, \tag{14}$$

and

$$\ln 2/w_k + \frac{1}{2} (\ln 2/w_k)^2 c_{i,k}^{ref} < (\widetilde{E}_i/d_i) \beta_k \left| h_{k,\min} \right|^2. \tag{15}$$

Since $c_{i,k}^{ref} > 0$ we have

$$\widetilde{E}_i > d_i \ln 2 / w_k \beta_k \left| h_{k,\min} \right|^2. \tag{16}$$

Thus, Theorem III.2 is proven. ■

E. The Reliability Model

In a heterogeneous fog computing system, the system reliability primarily comprises two components: channel transmission reliability for wireless communication between the vehicle and MEC, and equipment computation reliability of the vehicle and MEC.

For the computation reliability, it is mostly related to the computational faults existing in the devices. In the studies related to embedded systems and cloud computing, there are two main temporal types of faults, i.e., the transient faults and the permanent faults, and only the former is considered in this paper. Transient failures are unpredictable and unavoidable for computing devices. Random hardware failures are typically modeled using a Poisson distribution [31].

Assume that λ_k is the constant failure rate of MEC$_k$ per unit time, then the computation reliability of the task t_i when executed on MEC$_k$ can be expressed as

$$R_{i,k}^{comp} = e^{-\lambda_k T_{i,k}^{comp}}, \tag{17}$$

where $T_{i,k}^{comp}$ is the computation time of task t_i executed on MEC$_k$, for all $0 \le k \le n$ [32].

In studies on URLLC and mobile cloud computing, communication reliability under fading channel systems is typically characterized by the communication outage rate [28], [33]. Interruption rate is an important performance indicator in wireless communication, and according to Shannon Channel Coding Theory, when the signal transmission rate is lower than the channel capacity, there must exist a coding technique that enables the signals to be received correctly. When the signal transmission rate exceeds the channel capacity, the signal cannot be received correctly, resulting in a communication interruption. Therefore, given the target transmission rate R and the channel capacity C, the communication outage rate can be defined as

$$P_{out} = \Pr(C < R). \tag{18}$$

According to the Shannon Capacity Formula, the channel capacity is monotonically increasing with the instantaneous signal-to-noise ratio. Assume that the actual received signal-to-noise ratio and the threshold signal-to-noise ratio at the communication receiver are SINR_r and SINR_0, respectively, the communication interruption rate can also be expressed as

$$P_{out} = \Pr(\text{SINR}_r < \text{SINR}_0). \tag{19}$$

When t_i is offloaded and executed remotely on MEC_k, according to (1), the received signal-to-noise ratio of the MEC_k is

$$\text{SINR}_r = P_{i,k}\beta_k|h_{i,k}|^2. \tag{20}$$

Suppose the threshold signal-to-noise ratio of MEC_k is $\gamma_{0,k}$. Then, according to (2) and (20), the communication interruption rate, in this case, can be expressed as

$$\begin{aligned} P_{i,k}^{out} &= \Pr(\text{SINR}_r < \gamma_{0,k}) \\ &= \Pr\left(|h_{i,k}|^2 < \frac{\gamma_{0,k}}{P_{i,k}\beta_k}\right) \\ &= \int_0^{\frac{\gamma_{0,k}}{P_{i,k}\beta_k}} \frac{1}{\sigma_k^2} e^{-\frac{x}{\sigma_k^2}} dx \\ &= 1 - exp\left\{-\frac{\gamma_{0,k}}{\beta_k P_{i,k}\sigma_k^2}\right\}. \end{aligned} \tag{21}$$

Accordingly, as task t_i is offloaded to MEC_k, the communication reliability can be obtained as

$$R_{i,k}^{comm} = 1 - P_{i,k}^{out} = exp\left\{-\frac{\gamma_{0,k}}{\beta_k P_{i,k}\sigma_k^2}\right\}, \tag{22}$$

for all $1 \le k \le n$. If t_i is executed locally on the vehicle, we consider the reliability $R_{i,0}^{comm} = 1$.

Since the communication process and computation process of the task are independent of each other, the reliability of t_i is

$$R_{i,k} = R_{i,k}^{comp} \cdot R_{i,k}^{comm}, 0 \le k \le n. \tag{23}$$

The execution of each task is also independent of each other, so the reliability of the vehicle application $\mathcal{A}$ is given by

$$R(\mathcal{A}) = \prod_{i=1}^{m} R_{i,k}, 0 \le k \le n. \tag{24}$$

IV. Problem Definition and Algorithm Formulation

A. Problem Definition

In this section, we will define and analyze the offloading problem examined in this study.

Consider a fog computing environment with one vehicle and n MECs. For the vehicle application $\mathcal{A}$, three key issues must be addressed. Firstly, it is essential to ascertain the commencement time for each task. These tasks are governed by a hierarchy of priority constraints, stipulating that a task can only be scheduled once all its preceding tasks have been successfully concluded. Secondly, a strategic decision must be made regarding the execution location for each task—whether it will be processed locally or delegated to a designated mobile edge computing platform. Lastly, beyond meeting the minimum energy requirements for the tasks, it is imperative to allocate surplus energy to ensure optimal system performance.

Based on the aforementioned analysis, we define the research problem addressed as follows: In a fog computing environment consisting of n fog node servers denoted as $M' = \{\text{MEC}_1, \text{MEC}_2, \ldots, \text{MEC}_n\}$, where MEC possess fixed computing speeds. Given a vehicle application $\mathcal{A} = (L, \prec)$ for a vehicle containing task set $L = (t_1, t_2, \ldots, t_m)$, $t_i = (r_i, d_i)$, $1 \le i \le m$. If $t_i \prec t_j$, then t_j must wait for t_i to finish executing before it can be executed. The energy consumed to execute the application cannot exceed the energy constraint $\widetilde{E}$, i.e., $E(\mathcal{A}) = \sum_{i=1}^{m} E_i \le \widetilde{E}$. Also, to ensure that the application operates properly, the energy constraint should be greater than the minimum scheduling energy, i.e., $\widetilde{E}(\mathcal{A}) \ge E_{\min}(\mathcal{A})$. The objective is to devise an energy allocation strategy that enhances application reliability while mitigating the excessive execution time overhead. Thus, the objective formulation can be expressed as

$$\max \quad R(\mathcal{A}) \tag{25a}$$

$$\min \quad T(\mathcal{A}) \tag{25b}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} E_i \le \widetilde{E} \tag{25c}$$

$$\widetilde{E} \ge E_{\min}(\mathcal{A}) \tag{25d}$$

B. Pre-Scheduling Algorithm

Before allocating energy to tasks, we need to determine their start times and the MEC where they will be executed. In fog computing, traditional task offloading algorithms (e.g., ECLL-H) [6] and ECLLTROS [34]) use Level-by-Level scheduling methods to eliminate priority constraints between tasks. However, such methods can result in servers remaining idle for extended periods, leading to low server utilization. In Section III-D, we demonstrate that it is possible to predict a task's execution time without exceeding its energy constraints. Therefore, we calculate the earliest start time (EST) and earliest finish time (EFT) for each task to ensure they can be executed as early as possible. The attributes $EST(t_i)$ and $EFT(t_i)$ represent the EST and EFT, respectively. In an application with priority constraints, tasks need to wait for all of their predecessor tasks

Algorithm 1: Pre-Scheduling Algorithm.

Input: $\mathcal{A} = (L, \prec)$, $\text{MEC}_k = (s_k, w_k, \beta_k, \lambda_k, h_{k,\min}, \gamma_{0,k})$, vehicle $= (\xi, \alpha, P_s, \lambda_0)$, and $\widetilde{E}$.
Output: A Task Offloading Strategy.
1: Calculate $E_{i,min}$ for all $1 \leq i \leq m$. Calculate $E_{\min}(\mathcal{A})$;
2: Push the entry tasks into the execution queue, $Queue.add(t_{entry})$;
3: **while** there are tasks that have not been executed **do**
4: $t_i \leftarrow Queue.poll()$;
5: Calculate $\widetilde{E_i}$ with average energy allocate method;
6: **for** $k = 0;\ k \leq n;\ k++$ **do**
7: Calculate $EFT(t_i, k)$ if $\widetilde{E_i} \geq E^*_{i,k}$;
8: **end for**
9: Offload t_i to MEC_x where $EFT(t_i, x) = \min_{0 \leq k \leq n} EFT(t_i, k)$;
10: **for** $t_x \in succ(t_i)$ **do**
11: $Queue.add(t_x)$ if all the $pred(t_x)$ are executed;
12: **end for**
13: **end while**

to be completed before they can be scheduled. In this paper, EST and EFT are relative to each server. We provide the formulas for calculating EST and EFT:

$$\begin{cases} EST(t_{entry}) = 0 \\ EST(t_{i,k}) = \max\{\text{MEC}_k^T, \max\limits_{t_x \in pred(t_i)} AFT(t_x)\}, \end{cases} \tag{26}$$

$$EFT(t_{i,k}) = EST(t_{i,k}) + T_{i,k}. \tag{27}$$

The term MEC_k^T represents the earliest time at which the MEC_k is ready to execute a task. $AFT(t_x)$ represents the actual finish time of t_x.

By using this method, we can offload tasks to the MEC with the shortest EFT. This not only improves the overall completion time of the application but also effectively reduces the idle time of each server. We refer to this step as pre-scheduling. The process of the pre-scheduling algorithm is shown in Algorithm 1. The specific steps of the algorithm are as follows:

1) In lines 1−2, perform initialization tasks, compute the minimum energy $E_{i,min}$ required for task execution, and finally, place the entry task into the queue of tasks awaiting execution.
2) In lines 4−5, extract a task from the queue and calculate its energy constraints $\widetilde{E_i}$. The allocated energy is calculated using the average energy allocation method.
3) In lines 6−9, calculate the EFT of the task on each server. It should be noted that $E_{i,min}$ can only ensure that one server can execute the task but cannot guarantee that all servers can execute the task. Therefore, if a server cannot execute the task, it should be disregarded.
4) In lines 10−12, we need to check whether, among all the successor tasks of the current task, there are those whose predecessor tasks have all been completed. We place these tasks into the queue of tasks awaiting execution.

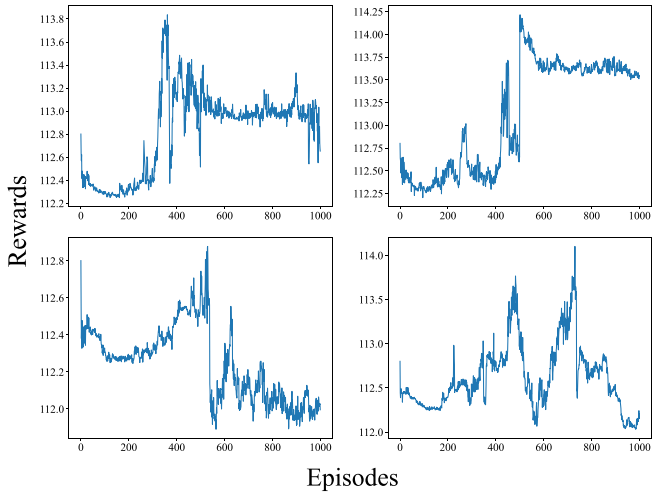

Fig. 5. Reward curves obtained using only PPO under four different reward function designs.

The time complexity of the pre-scheduling algorithm is analyzed as follows. First, in line 1, it takes $O(m)$ time to calculate the minimum energy for each task. In lines 6−8, we calculate the EFT of the task on all servers, with a time complexity of $O(m)$. In the while loop from lines 3−12, we iterate over all tasks, resulting in a complexity of $O(m)$. Therefore, the overall time complexity of Algorithm 1 is $O(m + mn)$.

In the pre-scheduling algorithm, we obtain the execution order of tasks and the servers on which the tasks are executed. This strategy is derived using the average energy allocation method. In Sections IV-C and IV-D, we will use reinforcement learning methods to reallocate the energy for tasks to enhance application reliability.

C. Generative Adversarial Imitation Learning and Proximal Policy Optimization

Reinforcement learning does not rely on labeled data like supervised learning, but it is highly dependent on the design of the reward function. Sometimes, even minor adjustments to the reward function can lead to significant differences in the training strategy. In many real-world scenarios, the reward function is not predefined, or the reward signals are extremely sparse. In such cases, randomly designing the reward function cannot ensure that the reinforcement learning-trained strategy meets actual needs, and the training may ultimately fail to converge. For example, in robotic arm control, the observation is the current state of the environment, and the action is how the arm should move next. If the reward is simply set to +1 for successfully grasping an object and -2 for failing to grasp, the agent might learn to avoid moving the arm to prevent negative rewards. Effective reward functions that aid in robotic arm control often require careful design and tuning by experts. Fig. 5 shows the reward curves obtained from four different reward functions, all of which performed poorly.

In reinforcement learning, imitation learning is a strategy for learning policies by observing and replicating expert behavior,

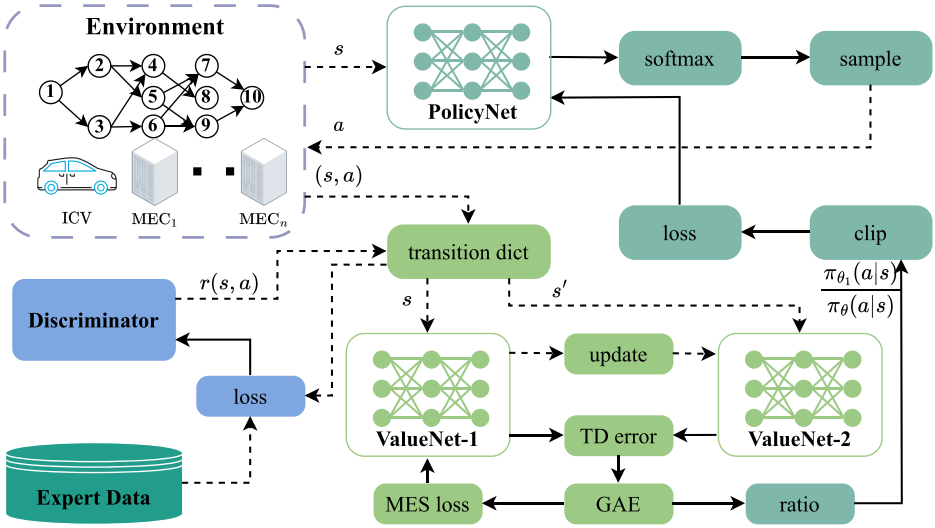

Fig. 6. The flow chart of generative adversarial imitation learning for Energy-Constrained Task Offloading.

thereby avoiding the need to train a policy from scratch. It learns to perform tasks from expert demonstrations, providing the learner with trajectory samples from the expert without allowing further queries to the expert during training and without providing any reinforcement signals. Currently, the main methods include behavior cloning (BC) [35], inverse reinforcement learning (IRL) [36], and GAIL [37], [38]. While BC is very simple, it only succeeds when dealing with large amounts of data due to compounding errors caused by covariate shift, and it is prone to overfitting in practice [37]. Many IRL algorithms are extremely expensive to run because they require reinforcement learning within their inner loops. Therefore, extending IRL methods to large-scale environments has been a recent focus of many studies [39]. GAIL, through adversarial training, can learn more complex and generalized policies that adapt to different state and action spaces. Compared to BC, GAIL does not require a large amount of labeled expert data; it can achieve effective learning with less data through adversarial training and significantly reduces the computational complexity of the algorithm. Most importantly, it does not need to provide explicit reward signals but relies on a discriminator to guide the generator in optimizing the policy.

GAIL is closely dependent on a Markov Decision Process (MDP). We need to model the problem as a MDP. To refine the operation, we divide the allocable energy ΔE into Km parts, where $\Delta E = \widetilde{E} - E_{\min}(\mathcal{A})$. Each time, one part of the energy is allocated to a task. In this way, we ensure that the energy distribution process for each task is carried out gradually, transforming it into a MDP, which is represented as

1) State: The state is the current energy allocation for each task and the reliability of the task.

2) Action: Action represents choosing which task to allocate energy to.

3) Reward: We generate the corresponding reward based on the output probability $\mathcal{D}(s,a)$ of the discriminator, which is calculated by the negative value of the logarithmic probability:

$$r(s,a) = -\log \mathcal{D}(s,a), \tag{28}$$

This reward mechanism gradually optimizes the agent's strategy through adversarial training, making the agent behave more and more like an expert, and finally can automatically learn high-quality strategies similar to the expert without explicit reward signals.

The GAIL training process mainly consists of 3 steps. The detailed flow chart of GAIL is shown in Fig. 6.

Step 1: Generate export data. The first step in imitation learning is to generate expert data. In this paper, we use the state and action sequences generated by a greedy algorithm as expert data. For the greedy algorithm, each time, the task with the highest overall reliability is selected for energy allocation. In other words, we evaluate the current reliability of each task and prioritize allocating energy to the more reliable tasks to maximize the probability of success in task execution. By choosing this greedy strategy, we were able to ensure that each energy allocation yielded optimal results, thus constructing a series of state-action trajectories that reflected the expert's behavior. These trajectories will be used as inputs for imitation learning for agents to learn and imitate during training.

Step 2: Sample trajectory. The agent samples trajectory data from the environment based on the current policy π, thereby generating state and action sequences. The discriminator $\mathcal{D}$ takes the state-action pair (s,a) as input and outputs a real number between 0 and 1, indicating whether (s,a) is from the agent's policy rather than the expert's policy.

Step 3: Optimize strategy. Agents optimize strategy through reinforcement learning algorithms. In this article, we use PPO to achieve this.

Let θ represent the parameters of the current policy network π_θ. We define the objective function as

$$J(\theta) = \mathbb{E}_{s_0}[V^{\pi_\theta}(s_0)] = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r(s,a)\right], \tag{29}$$

where s_0 represents the initial state, and γ represents the discount factor. We consider how to use the current θ to find a better parameter θ_1, such that $J(\theta_1) \geq J(\theta)$. Since the objective

function is independent of the initial state s_0 distribution and policy, $J(\theta)$ can be rewritten as

$$J(\theta) = \mathbb{E}_{s_0}[V^{\pi_\theta}(s_0)] = -\mathbb{E}_{\pi_{\theta_1}}\left[\sum_{t=0}^{\infty}\gamma^t\left(\gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right)\right]. \quad (30)$$

At this point, we can obtain the difference between the objective functions of the new and old policies as

$$J(\theta_1) - J(\theta) = \mathbb{E}_{s_0}\left[V^{\pi_{\theta_1}}(s_0)\right] - \mathbb{E}_{s_0}\left[V^{\pi_\theta}(s_0)\right] = \mathbb{E}_{\pi_{\theta_1}}\left[\sum_{t=0}^{\infty}\gamma^t\left[r(s_t,a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right]\right]. \quad (31)$$

We let $A_t^\theta = r(s_t,a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$, and define the state visitation distribution $\nu^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P_t^\pi(s)$, where $P_t^\pi(s)$ represents the probability of the agent being in state s at time t under policy π. Now

$$J(\theta_1) - J(\theta) = \mathbb{E}_{\pi_{\theta_1}}\left[\sum_{t=0}^{\infty}\gamma^t A_t^\theta\right] = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\nu^{\pi_{\theta_1}}}\mathbb{E}_{a\sim\pi_{\theta_1}(\cdot|s)}\left[A_t^\theta\right]. \quad (32)$$

Therefore, as long as we can find a new policy such that $\mathbb{E}_{s\sim\nu^{\pi_{\theta_1}}}\mathbb{E}_{a\sim\pi_{\theta_1}(\cdot|s)}[A_t^\theta] \geq 0$, we can ensure that the policy performance is monotonically increasing. For A, we use a common method known as Generalized Advantage Estimation (GAE). The key function of GAE is

$$A_t^{GAE} = \sum_{l=0}^{\infty}(\gamma\kappa)^l\delta_{t+l}, \quad (33)$$

where δ_t is the temporal difference error of the state value and $\kappa \in [0,1]$ is the decay factor. However, directly solving (32) remains challenging because π_{θ_1} is the policy we need to solve for, yet we also need to use it to collect samples. PPO makes an approximation at this step; specifically, it ignores the change in state visitation distribution between the two policies and directly uses the state distribution of the old policy. Thus, (32) can be rewritten as

$$J(\theta_1) - J(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\nu^{\pi_{\theta_1}}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[r_t(\theta)A_t^\theta\right], \quad (34)$$

where

$$r_t(\theta) = \pi_{\theta_1}(a|s)/\pi_\theta(a|s). \quad (35)$$

At this point, since we approximate the state visitation distributions of the two policies as the same, we need to ensure $\mathbb{E}_{s\sim\nu^{\pi_\theta}}[D_{KL}(\pi_\theta(\cdot|s),\pi_{\theta_1}(\cdot|s))] \leq \psi$. Additionally, PPO employs the (PPO-clip) method to restrict policy updates, ensuring that the difference between the new parameters and the old parameters is not too large. This is represented by

$$\mathcal{L}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right], \quad (36)$$

where $clip(x,y,z) := \max(\min(x,z),y)$. It limits the policy ratio to the range $[1-\epsilon, 1+\epsilon]$, with ϵ being a small clipping hyperparameter.

Next, we optimize the discriminator in GAIL. First, we obtain state-action sequences from expert data as positive samples, and then use trajectory data generated by the current generator as negative samples. The essence of GAIL is to ensure that the occupancy measure $\rho_\pi(s,a)$ of state-action pairs (s,a) under the current policy of the agent aligns with that of the expert policy, where $\rho^\pi(s,a) = \nu^\pi(s,a)\pi(a|s)$, representing the probability that a state-action pair (s,a) is visited. Therefore, the goal of the discriminator $\mathcal{D}$ is to make the output for expert data close to 0 and the output for the imitator's policy close to 1. We use the cross-entropy loss function to measure the discriminator's accuracy on positive and negative samples, with the loss function given by

$$\mathcal{L}(\phi) = -\mathbb{E}_{\rho^\pi}[\log\mathcal{D}_\phi(s,a)] - \mathbb{E}_{\rho^E}[\log(1-\mathcal{D}_\phi(s,a))], \quad (37)$$

where ϕ is the parameters of the discriminator $\mathcal{D}$. The loss function is minimized through backpropagation and optimization algorithms (such as Adam). Finally, the generator and discriminator are optimized alternately until the generator's policy stabilizes and the discriminator can no longer effectively distinguish between generated data and expert data (i.e., the discriminator's output approaches 0.5, indicating that generated data and expert data are indistinguishable). Through continuous adversarial training, the distribution of data generated by the imitator's policy will approach the real expert data distribution, achieving the goal of imitation learning.

D. Reliability-Enhanced GAIL for Energy-Constrained Task Offloading Algorithm

In Sections IV-B and IV-C, we introduced the pre-scheduling algorithm and the implementation principles of GAIL. In this section, we will elaborate on the specific implementation process of reliability-enhanced GAIL for energy-constrained task offloading algorithm (REGAIL), as shown in Algorithm 2. The specific steps of the algorithm are as follows:

1) In lines 1−4, Initialize the parameters and data required for the algorithm.
2) In lines 6−11, at each iteration, extract an energy portion of size $\Delta E/(Km)$ from the remaining energy, allocate it to a task according to the policy π_θ, and calculate the reward obtained. Repeat this process until the remaining energy is depleted.
3) In lines 12−13, obtain the reliability and execution time of the application under the new policy, and retain the policy with the higher reliability. Add the state-action pairs generated under this policy to the expert dataset, enriching the dataset.
4) In lines 16−21, Update the policy network and discriminator network.

The time complexity of the REGAIL is analyzed as follows. In line 2, the time complexity of Algorithm 1 is $O(m+nm)$. From lines 6 to 11, the operations are performed Km times, resulting in a time complexity of $O(Km)$. The network updating from lines 12 to 19 depends on factors such as network architecture, batch size, and optimization algorithm, so we do not compute

Algorithm 2: Reliability-Enhanced GAIL for Energy-Constrained Task Offloading Algorithm (REGAIL).

Input: $\mathcal{A} = (L, \prec)$, $\text{MEC}_k = (s_k, w_k, \beta_k, \lambda_k, h_{k,\min}, \gamma_{0,k})$, vehicle $= (\xi, \alpha, P_s, \lambda_0)$, and $\widetilde{E}$.
Output: $T(\mathcal{A})$ and $R(\mathcal{A})$.

```
T(A) ← 0, R(A) ← 1;
Obtain the offloading strategy S and generate expert data using Algorithm 1.
Initialize the generator policy network π_θ and the discriminator network D_φ.
for (iter = 1 → epoch) do
    while (Remaining energy E^rem > 0) do
        Select a task t_i based on the policy π_θ, and allocate ΔE/(Km) energy;
        Add the current (s_t, a_t) to the generator's policy set;
        Calculate the reward r(s_t, a_t) using Eq. (28);
        E^rem = E^rem − ΔE/(Km);
    end while
    Obtain the reliability R and execution time T for the current task offloading.
    if R > R(A) then
        T(A) ← T, R(A) ← R;
        Store the state-action pairs into the expert dataset.
    end if
    Construct the training set by labeling the expert data as positive samples (label 1) and the generated data as negative samples (label 0);
    Calculate discriminator's loss function using Eq. (37);
    Update the parameters φ using the Adam optimizer;
    Calculate A_t^θ and r_t(θ) using Eqs. (34) and (35);
    Calculate policy network's loss function using Eq. (36);
    Update the parameters θ using the Adam optimizer;
end for
Return R(A), T(A).
```

the complexity here. Therefore, the overall time complexity of REGAIL is $O((1 + n + K \cdot epoch)m)$.

V. Experiment Performance Evaluation

In this section, simulation experiments based on the Python programming language are conducted to evaluate the performance of REGAIL in this paper. The experiments were conducted on a Windows 11 operating system, with an 11th Gen Intel(R) Core(TM) i5-13600kF CPU running at 2.60 GHz. The computer had 16.0 GB of memory, of which 15.8 GB was available for use. The graphics card used was an NVIDIA RTX4070 Super. The Python version used is 3.9, with Torch version 1.13.1, and cudnn version 11.6.

A. Experiment Setting

A simple fog computing environment is examined, consisting of one vehicle and seven MECs. The vehicle is configured with the following parameters: $\xi = 0.1, \alpha = 2.0, P_s = 0.05\,\text{W}, \lambda_0 = 0.01$. The MEC_k is configured with the following parameters: $s_k = 20.1 + 0.1k$BI/s, $w_k = 2.9 + 0.1k$MB/s, $\beta_k = 5.4 - 0.2k\text{W}^{-1}$, $\lambda_k = 0.0015 + 0.0001k$, for all $1 \le k \le n$. The settings of these parameters remain mostly consistent with previous studies [6], [30], [40]. In addition, the rayleigh random fading model is introduced. According to the analysis in Section III-B, it is clear that the channel power fading factor $|h_{i,k}|^2$, which communicates between the vehicle and the MEC_k during task scheduling, obeys an exponential distribution with mean σ_k^2, and is a continuous random variable greater than zero. To simplify the problem analysis and experiments, we assume that the channel power fading factor $|h_{i,k}|^2$ takes a series of fixed discrete values in the actual scheduling process, i.e., given the $|h_{i,k}|^2 \in \{|h_{\min}|^2, |h_1|^2, |h_2|^2, \ldots, |h_{\max}|^2\}$, where the same interval $\Delta|h|^2$ is taken between each discrete value. Firstly, the reference probability of each discrete value is taken according to the mean value σ_k^2:

$$P_{ref}(|h_{i,k}|^2 = |h_j|^2) = P(|h_j|^2 - \Delta|h|^2 < x < |h_j|^2) = \int_{|h_j|^2-\Delta|h|^2}^{|h_j|^2} \frac{1}{\sigma_k^2} e^{-\frac{x}{\sigma_k^2}} dx. \tag{38}$$

Then take the total reference probability as

$$P_{k,sum} = \sum_j P_{ref}(|h_{i,k}|^2 = |h_j|^2). \tag{39}$$

Finally, the probability of taking each discrete value is obtained as

$$P(|h_{i,k}|^2 = |h_j|^2) = \frac{P_{ref}(|h_{i,k}|^2 = |h_j|^2)}{P_{k,sum}}. \tag{40}$$

In particular, the MEC_k has factor $|h_j|^2$ to fall into $|h_{i,k}|^2 \in \{0.4, 0.6, 0.8, 1.0\}$, according to section III-D, $h_{k,min}$ is a known parameter for vehicle, obviously taken $|h_{k,\min}|^2 = 0.4$ here, and the fading mean σ_k^2 is set to obey independent uniform distribution within the interval [0.5,0.9], and the signal-to-noise ratio threshold is $\gamma_{0,k} = 1.0 + 2.0k$, for all $1 \le k \le n, 1 \le i \le m$.

This study uses randomly generated DAGs to simulate different vehicle applications. We generate a random DAG using the *G(m, p)* [41] method, where $p = 0.45$. For each task in application $\mathcal{A}$, its computation and communication requirements are generated randomly, i.e., for task t_i, the r_i obeys independent uniform identical distribution in the range $[10.0, 14.0]$ and the d_i obeys independent uniform identical distribution in the range $[3.0, 5.0]$, where $1 \le i \le m$.

B. Comparative Experiment With Different K Value

In the REGAIL algorithm, the remaining energy is divided into Km portions, with one task selected at a time to receive one portion of this energy. In this section, we will specifically discuss the selection of the hyperparameter K. We set the number of tasks $m = 20$ and vary K from 1 to 10 (to avoid excessive training complexity due to large K values), training a total of 10,000 episodes. The reward curve, execution time curve, and reliability curve for these experiments are shown in Figs. 7–8,

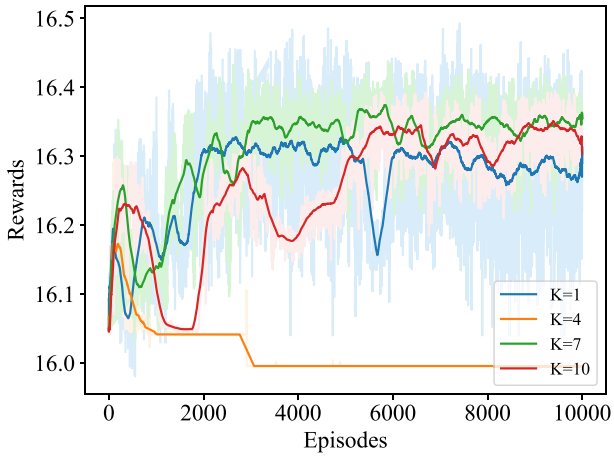

Fig. 7. The reward curve graphs for different K values when $m = 20$.

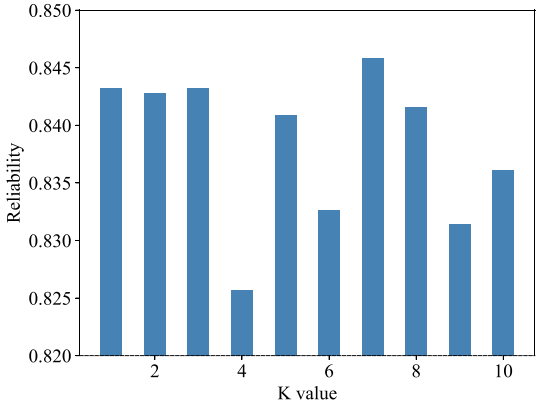

Fig. 8. The bar chart of application reliability for different K values when $m = 20$.

respectively. To simplify the presentation, we only show the reward curves for $K = 1, 4, 7$, and 10.

Observing Fig. 7, we can see that the reward curves for $K = 1, 7, 10$ gradually stabilize as training progresses. However, for $K = 4$, the curve initially rises but then suddenly drops and remains fixed at 15.9, indicating the training performance is poor, and the negative effects of hyperparameters are accumulated, breaking the stability of learning. After 4000 iterations, the reward curve is close to a straight line, indicating that the model has completely lost its learning ability and has fallen into a state of invalid loop. At $K = 1$, the reward initially increases rapidly, but the curve exhibits high volatility and instability, with sudden drops. For $K = 10$, the reward curve eventually converges to a satisfactory level, but the larger K value leads to longer training times.

Overall, $K = 7$ provides the most balanced performance. The reward gradually increases and stabilizes without significant fluctuations, and the training time is not excessively long. Additionally, as shown in Figs. 8 and 9, after reinforcement learning, the application's reliability and completion time achieve optimal values when $K = 7$. Therefore, in subsequent experiments, we will set K to 7.

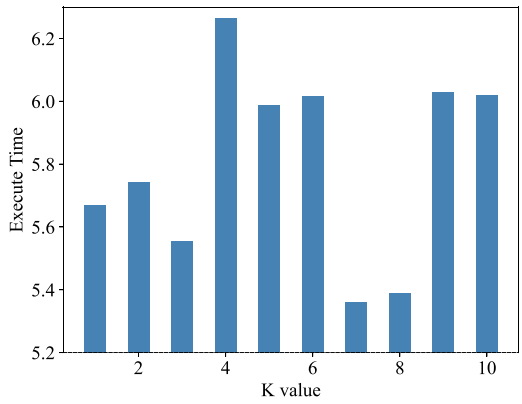

Fig. 9. The bar chart of application execute times for different K values when $m = 20$.

TABLE III
RELIABILITY EXPERIMENTAL DATA OF DIFFERENT REINFORCEMENT LEARNING ALGORITHMS COMBINED WITH GAIL UNDER DIFFERENT NUMBER OF TASKS

	20	40	60	80
TRPO-GAIL	0.8374	0.7233	0.6194	0.5532
SAC-GAIL	0.8458	0.7151	0.6329	0.5441
REGAI L	**0.8479**	**0.7362**	**0.6391**	**0.5571**

C. Comparative Experiments Combining GAIL With Different Reinforcement Learning Algorithms

In REGAIL, we use PPO as the policy optimization algorithm. In this section, we will compare the performance of GAIL combined with various policy-based reinforcement learning algorithms. The algorithms included in the comparison are as follows:

- TRPO (Trust Region Policy Optimization) is known for its ability to provide reliable policy updates by ensuring that each update remains within a safe trust region. This helps to avoid large, potentially harmful updates that could degrade performance [42].
- SAC (Soft Actor-Critic) is a popular off-policy algorithm that aims to maximize both the reward and the entropy of the policy, encouraging exploration and leading to more robust policies [43].

Fig. 10 shows the reward curves for three algorithms — TRPO-GAIL, SAC-GAIL, and REGAIL — across different task numbers ($m = 20$, 40, 60, and 80). It is observed that TRPO-GAIL exhibits the most stable training results with smooth curves, though it does not achieve the highest rewards, consistently falling below REGAIL. SAC-GAIL, on the other hand, shows noticeable fluctuations in its reward curve and also does not reach high final rewards. Overall, REGAIL's reward curves are generally higher than those of TRPO-GAIL and SAC-GAIL, except for $m = 60$, and the curves converge more smoothly over time.

Tables III and IV provide experimental data on reliability and execution time for different reinforcement learning algorithms combined with GAIL across various task numbers. Analysis of the data reveals that while the reliability and execution times

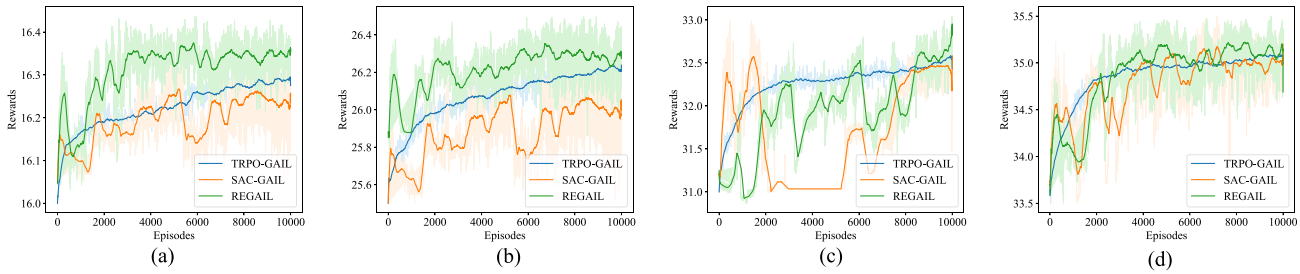

Fig. 10. Reward curves for different reinforcement learning algorithms combined with GAIL across varying task numbers. (a) $m = 20$. (b) $m = 40$. (c) $m = 60$. (d) $m = 80$.

TABLE IV
EXECUTE TIME EXPERIMENTAL DATA OF DIFFERENT REINFORCEMENT LEARNING ALGORITHMS COMBINED WITH GAIL UNDER DIFFERENT NUMBER OF TASKS

	20	40	60	80
TRPO-GAIL	6.011	**15.031**	23.695	35.671
SAC-GAIL	5.926	16.022	24.842	35.892
REGAI L	**5.360**	15.316	**23.622**	**35.323**

of the three algorithms are quite close, REGAIL consistently achieves better results. Specifically, REGAIL delivers the highest reliability at $m = 40$, 60, and 80, and the shortest execution time at $m = 20$, 60, and 80.

In summary, the REGAIL, which combines PPO with GAIL, not only provides more stable training results but also excels in terms of reliability and execution time compared to other reinforcement learning algorithms.

D. Comparison of Reliability and Execution Time Between REGAIL and Other Algorithms Under Different Number of Tasks

The primary goal of REGAIL is to significantly improve reliability while ensuring that the increase in execution time remains minimal. In this section, we will compare the execution time and reliability of the REGAIL algorithm with other existing algorithms. These algorithms include the following two:

- Greedy algorithm selects the task that maximizes overall reliability at each step of energy allocation. This is the expert data used in our study.
- ECLL (Energy-Constrained Level-by-Level Scheduling) algorithm, designed by Li [6], is a hierarchical energy-constrained task offloading algorithm. It is considered a classic approach in the field of fog computing for task offloading.

Figs. 11 and 12 illustrate the application reliability and execution time for the three algorithms under varying task quantities. The experimental results show that REGAIL consistently outperforms the other algorithms in terms of reliability, even surpassing the Greedy algorithm. This indicates that REGAIL, through adversarial training, can correct suboptimal behaviors in the expert strategy, yielding superior results. For instance, at $m = 60$, REGAIL's reliability exceeds that of Greedy by 5.85% , marking its highest performance. ECLL, not primarily

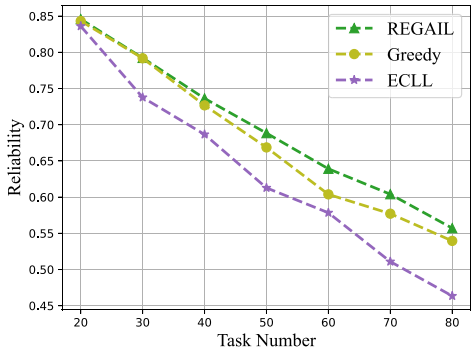

Fig. 11. Line chart of the reliability of different algorithms under different number of tasks.

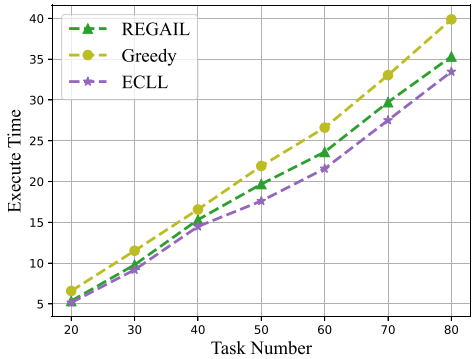

Fig. 12. Line chart of the execute time of different algorithms under different number of tasks.

designed with reliability in mind, exhibits the poorest reliability compared to REGAIL and Greedy. However, ECLL achieves the fastest execution times across all task quantities, followed by REGAIL, with Greedy being the slowest. When considering both reliability and execution time, REGAIL shows a 20.19% higher reliability than ECLL at $m = 80$, with a trade-off of 5.56% slower execution time. On average, REGAIL's reliability is 11.01% higher than ECLL's, while its execution time is slower by 7.30% .

E. Experiment Summary

By comparing the proposed GAIL with other reinforcement learning algorithms (TRPO, SAC), the results show that the combination of GAIL and PPO is the most powerful and effective,

and the reward curves of REGAIL on different task amounts show excellent convergence. Meanwhile, the comparative experiments of the proposed REGAIL with Greedy and ECLL algorithms show that REGAIL not only outperforms Greedy in terms of reliability, but also achieves a good balance between reliability and execution time. Compared with ECLL, REGAIL significantly improves reliability while maintaining control over execution time.

VI. Conclusion

To improve the reliability of ICV applications, this paper designs a reliability model suitable for fog computing conditions and incorporates hardware failure rate and communication interruption rate. This paper also proves the effectiveness of using the minimum fading coefficient to predict the channel transmission time to ensure that it meets the energy constraint. On this basis, we propose the REGAIL algorithm. The REGAIL algorithm first uses a pre-scheduling method to determine whether the task needs to be offloaded and which server to execute it. Then, the algorithm combines GAIL and PPO and uses a greedy algorithm as expert data to train the energy allocation strategy to improve the reliability of the application. By integrating GAIL, we solve the instability problem in PPO training and correct the suboptimal behavior in the expert strategy through adversarial training, so that potential better strategies can be explored in a wider state space. In addition, GAIL continuously improves the generator strategy during training to adapt to different environmental changes and complex scenarios, and sometimes even surpasses the performance of the expert strategy. Experimental results show that our algorithm achieves a good balance between reliability and execution time. For example, compared with ECLL, the proposed REGAIL algorithm significantly improves reliability while maintaining control over execution time. Future research will explore the integration of multi-modal learning and aggregation methods to enhance task offloading efficiency for intelligent connected vehicles in fog computing systems.

References

[1] W. Zhan et al., "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[2] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[3] X. An, R. Fan, H. Hu, N. Zhang, S. Atapattu, and T. A. Tsiftsis, "Joint task offloading and resource allocation for IoT edge computing with sequential task dependency," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16546–16561, Sep. 2022.

[4] G. Xie et al., "Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems," *IEEE Trans. Serv. Comput.*, vol. 13, no. 5, pp. 871–886, Sep./Oct. 2020.

[5] L. Zhao, Y. Ren, Y. Xiang, and K. Sakurai, "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems," in *Proc. IEEE 12th Int. Conf. High Perform. Comput. Commun.*, 2010, pp. 434–441.

[6] K. Li, "Scheduling precedence constrained tasks for mobile applications in fog computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 2153–2164, May/Jun. 2023.

[7] N. Kumari, A. Yadav, and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," *Comput. Netw.*, vol. 214, 2022, Art. no. 109137.

[8] J. Liang, K. Li, C. Liu, and K. Li, "Joint offloading and scheduling decisions for dag applications in mobile edge computing," *Neurocomputing*, vol. 424, pp. 160–171, 2021.

[9] R. Buyya and S.N. Srirama, *Fog and Edge Computing: Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2019.

[10] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, early access, Mar. 12, 2019, doi: 10.1109/TSUSC.2019.2904680. [Online]. Available: https://ieeexplore.ieee.org/document/8665964/

[11] J. Liu, S. Wang, J. Wang, C. Liu, and Y. Yan, "A task oriented computation offloading algorithm for intelligent vehicle network with mobile edge computing," *IEEE Access*, vol. 7, pp. 180491–180502, 2019.

[12] N. Sharma, A. Ghosh, R. Misra, and S. K. Das, "Deep meta Q-learning based multi-task offloading in edge-cloud systems," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2583–2598, Apr. 2024.

[13] W. Fan, Y. Yu, C. Bao, and Y. Liu, "Vehicular edge intelligence: DRL-based resource orchestration for task inference in vehicle-RSU-edge collaborative networks," *IEEE Trans. Mobile Comput.*, vol. 24, no. 10, pp. 10927–10944, Oct. 2025.

[14] F. Alqahtani, M. Amoon, and A. A. Nasr, "Reliable scheduling and load balancing for requests in cloud-fog computing," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 4, pp. 1905–1916, Jul. 2021.

[15] R. Liu et al., "Reliability optimization scheduling and energy balancing for real-time application in fog computing environment," in *Proc. 15th Int. Symp. Adv. Parallel Process. Technol.*, C. Li, Z. Li, L. Shen, F. Wu, and X. Gong, Eds., Singapore: Springer, 2024, pp. 180–200.

[16] S. M. Azimi, O. Simeone, O. Sahin, and P. Popovski, "Ultra-reliable cloud mobile computing with service composition and superposition coding," in *Proc. 2016 Annu. Conf. Inf. Sci. Syst.*, 2016, pp. 442–447.

[17] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.

[18] Y. Shang, J. Li, and X. Wu, "Dag-based task scheduling in mobile edge computing," in *Proc. 7th Int. Conf. Inf. Sci. Control Eng.*, 2020, pp. 426–431.

[19] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[20] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[21] K. -L. Besser and E. A. Jorswieck, "Reliability bounds for dependent fading wireless channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 5833–5845, Sep. 2020.

[22] P. Dai, Y. Huang, K. Hu, X. Wu, H. Xing, and Z. Yu, "Meta reinforcement learning for multi-task offloading in vehicular edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2123–2138, Mar. 2024.

[23] Z. Wei, B. Li, R. Zhang, X. Cheng, and L. Yang, "Many-to-many task offloading in vehicular fog computing: A multi-agent deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2107–2122, Mar. 2024.

[24] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: A matching-theoretic framework," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 100–106, Sep. 2019.

[25] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 2041–2057, Feb. 2022.

[26] C. Li, Y. Zhang, and Y. Luo, "A federated learning-based edge caching approach for mobile edge computing-enabled intelligent connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3360–3369, Mar. 2023.

[27] A. Zhadan et al., "Multi-agent reinforcement learning-based adaptive heterogeneous dag scheduling," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 5, pp. 1–26, 2023.

[28] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2062–2075, Sep. 2020.

[29] J. N. Laneman, D. N. Tse, and G. W. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *IEEE Trans. Inf. theory*, vol. 50, no. 12, pp. 3062–3080, Dec. 2004.

[30] K. Li, "Heuristic computation offloading algorithms for mobile users in fog computing," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 2, pp. 1–28, 2021.

[31] J. Peng, K. Li, J. Chen, and K. Li, "Reliability/performance-aware scheduling for parallel applications with energy constraints on heterogeneous computing systems," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 3, pp. 681–695, Jul.–Sep. 2022.
[32] H. Liu, L. Cao, T. Pei, Q. Deng, and J. Zhu, "A fast algorithm for energy-saving offloading with reliability and latency requirements in multi-access edge computing," *IEEE Access*, vol. 8, pp. 151–161, 2020.
[33] H. -C. Yang, S. Choi, and M. -S. Alouini, "Ultra-reliable low-latency transmission of small data over fading channels: A data-oriented analysis," *IEEE Commun. Lett.*, vol. 24, no. 3, pp. 515–519, Mar. 2020.
[34] R. Liu, W. Wu, X. Guo, G. Zeng, and K. Li, "Replica fault-tolerant scheduling with time guarantee under energy constraint in fog computing," *Future Gener. Comput. Syst.*, vol. 159, pp. 567–579, 2024.
[35] U. Syed, M. Bowling, and R. E. Schapire, "Apprenticeship learning using linear programming," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1032–1039.
[36] B. D. Ziebart et al., "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, Chicago, IL, USA, 2008, vol. 8, pp. 1433–1438.
[37] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 1–9.
[38] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 1–12.
[39] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 49–58.
[40] K. Li, "How to stabilize a competitive mobile edge computing environment: A game theoretic approach," *IEEE Access*, vol. 7, pp. 69960–69985, 2019.
[41] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J. -M. Vincent, and F. Wagner, "Random graph generation for scheduling simulations," in *Proc. 3rd Int. ICST Conf. Simul. Tools Techn.*, 2010, pp. 1–10.
[42] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[43] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

Wufei Wu (Senior Member, IEEE) received the Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, in 2018. He was a Visiting Scholar with Nagoya University, Nagoya, Japan, from 2016 to 2017. He is currently an Associate Professor with the School of Information Engineering, Nanchang University, Nanchang, China. He has worked on the scheduling optimization for distributed systems, embedded real-time system, and cybersecurity enhancements for in-vehicle networks. He is a Senior Member of CCF.

Jia Wei is currently working toward the M.S. degree in electronics and communication engineering with Nanchang University, Nanchang, China. Her research focuses on parallel and distributed computing.

Xiaochuan Guo received the master's degree from Nanchang University, Nanchang, China. He is currently working toward the doctoral degree with the School of Information Science and Engineering, Hunan University, Changsha, China. His research interests include fog computing and multi-objective optimization.

Jiaxin Zeng is currently working toward the M.S. degree with the School of Information Engineering, Nanchang University, Nanchang, China. Her research interests include heterogeneous distributed system, edge computing, and embedded system.

Gang Zeng (Member, IEEE) received the Ph.D. degree in information science from Chiba University, Chiba, Japan, in 2006. From 2006 to 2010, he was a Researcher and then an Assistant Professor with the Center for Embedded Computing Systems, Graduate School of Informatics, Nagoya University, Nagoya, Japan, where he is currently an Associate Professor with the Graduate School of Engineering. His research interests include power-aware computing and real time embedded system design. He is a member of IEICE, IPSJ, and JSAE.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990. He is currently SUNY Distinguished Professor with the State University of New York, Albany, NY, USA, and a National Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored more than 1200 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top few most influential Scientists in parallel and distributed computing, regarding single-year impact (ranked #2) and career-long impact (ranked #3) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars from 2023 to 2025 and is among the top 0.02% out of more than 20 million scholars worldwide based on top-cited publications in the last ten years. He is listed in ScholarGPS Highly Ranked Scholars (2022-2024) and is among the top 0.002% out of more than 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He was the recipient of the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was the recipient of the 2022-2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS Fellow, AAIA Fellow, ACIS Fellow, and AIIA Fellow. He is a member of the European Academy of Sciences and Arts. He is a member of Academia Europaea (Academician of the Academy of Europe).