A Multi-tier Offloading Optimization Strategy for Consumer Electronics in Vehicular Edge Computing

Haiyang Lin, Bo Xiao, Xiaokang Zhou, Yonghong Zhang, Xiaodong Liu

***Abstract*—In the domain of consumer electronics, vehicular edge computing (VEC) technology is emerging as a novel data processing paradigm within vehicular networks. By sending tasks related to vehicular applications to the edge, this model makes it easier for computing power to be spread out. This lets interactive services respond quickly. Nevertheless, the computational resources at edge servers are inherently limited and often tasked with handling multiple concurrent operations. The inefficacious allocation of these resources significantly impairs the efficiency of task offloading. Additionally, indiscriminate offloading could overwhelm the servers, detrimentally impacting the performance of subsequent tasks. To circumvent these challenges, this study introduces a multi-tier offloading model predicated on game theory principles. This framework aims to optimize resource utilization at the edge while accounting for server load to ensure the timely execution of latency-sensitive tasks. To evaluate this model, this paper created a simulation environment specifically for video game tasks in consumer electronics. The experimental results show that the multi-tier offloading model can effectively relieve the load pressure on the edge server. The task failure rate of the multi-tier offloading model remains at the lowest level compared with several state-of-the-art algorithms, significantly reducing the execution delay of tasks and being able to meet the requirements of consumer electronics applications.**

***Index Terms*—Vehicular Edge Computing; Resource Allocation; Task Offloading**

I. Introduction

A. Background

As 5G communication technology and the Internet of Vehicles advance, a multitude of vehicular network applications have emerged, including self-driving vehicles, live traffic updates, and virtual reality experiences within vehicles [1], [2]. To safeguard drivers and passengers and improve the driving experience, these applications necessitate the effective handling of vast amounts of computational data [3]. The computational and storage resources of individual vehicle devices are limited and cannot support the processing of such compute-intensive applications, highlighting the need for collaborative work among multiple node resources [4]. Cloud computing is a centralized resource pool sharing model [5], requiring data in the internet of vehicles to be transmitted over long distances to cloud centers for processing and then fed back to edge devices [6]. The inevitable network fluctuations and transmission interruptions during data transmission increase security risks and affect service quality [7]. VEC, by deploying small data centers at the network edge, provides computing services to vehicles near the data terminals, effectively reducing reliance on public network transmission, fully utilizing edge device resources, and enhancing the distributed processing capability of the overall network resources, making it a suitable solution for the Internet of Vehicles scenario [8].

The development of VEC technology provides convenience for addressing the strict communication requirements of compute-intensive vehicular application tasks [9]. Edge servers have a portion of the computing power of cloud servers and are closer to vehicle terminals [10]. Vehicles no longer need to obtain computation offloading services from remote cloud data centers but can rely on the computing and caching resources provided by edge servers deployed at the vehicle access network side to enhance the processing capabilities of terminal devices significantly [11]. The shorter transmission distances provided by VEC technology can bring lower propagation, computation, and communication delays. This also helps reduce the risk of round-trip network congestion, meeting the low latency and ultra-reliability requirements of vehicular application tasks. However, in the face of growing service applications in the VEC, edge servers have limited computing resources and cannot accommodate all vehicles for offloading at the same time [12]. The computing and storage capabilities of edge servers cannot match those of cloud servers [13], and blindly offloading computational tasks under the conditions of limited edge server resources can easily lead to competition for communication, computation, and storage resources among different vehicles [14], resulting in substantial waiting delays and resource wastage, thereby affecting the execution efficiency of tasks and even possibly failing to meet the normal computational needs of some applications [15].

In recent years, many scholars have studied task offloading, but there are also some limitations. In some works, a scheduling center is required to implement some functions of task offloading (such as formulating offloading strategies and resource allocation) [16]. However, when the number of vehicles and the volume of vehicle requests increase, centralized

This work has received funding from National Natural Science Foundation of China (No. 42275157). *(Haiyang Lin and Bo Xiao contributed equally to the work.) (Corresponding author: Xiaokang Zhou.)*

Haiyang Lin is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: 202212490371@nuist.edu.cn).

Bo Xiao is with Jiangsu Province Engineering Research Center of Advanced Computing and Intelligent Services, Nanjing 210044, China, and also with the School of Chemistry and Materials Science, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: 002479@nuist.edu.cn).

Xiaokang Zhou is with the Faculty of Business Data Science, Kansai University, Osaka 565-0823, Japan (e-mail: zhou@kansai-u.ac.jp).

Yonghong Zhang is with the School of Automation, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: zyh@nuist.edu.cn).

Xiaodong Liu is with the School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, U.K. (e-mail: x.liu@napier.ac.uk).

facilities will experience a sharp increase in problem scale, leading to slower processing efficiency, increased vehicle waiting delays, and increased device energy consumption [17]. Moreover, centralized architectures are prone to single points of failure, which can affect the performance of the entire system and increase vehicle waiting delays. Some works only focus on optimizing offloading decisions and use edge server resources extensively. When the number of vehicles increases, edge servers will experience continuous overload, affecting the execution of subsequent tasks [18].

B. Solutions and Contributions

This study delves into the challenge of task computation offloading for multiple devices in the context of VEC. To optimize the distribution of resources at edge servers, game theory is employed to formulate the task offloading problem as a non-cooperative multi-vehicle computation offloading game, and a multi-tier offloading model is developed. The specific contributions are as follows:

1) A pricing scheme that considers the workload of edge servers is proposed. The real-time processing capability of an edge server is related to its workload. Vehicles should pay proportionally based on the remaining computing resources of the edge server, the vehicle's CPU resource usage, and the amount of data offloaded to alleviate the overload problem of edge servers.
2) The offloading of tasks is defined in two stages: the dynamic game offloading stage and the equilibrium offloading stage. The dynamic game offloading stage refines the use of edge server resources by formulating the task offloading problem as a Stackelberg game. In the equilibrium offloading stage, edge servers recommend the server resources and the quantity of data to be offloaded based on the server's workload and the latency constraints of the tasks, guiding vehicles in task offloading and reducing the failure rate of tasks.
3) This paper constructs a simulation environment based on the characteristics of video game tasks in the consumer electronics field. Experimental comparisons with existing algorithms show that the proposed multi-tier offloading model can reasonably use the resources of edge servers and meet the execution requirements of tasks.

In this paper, related work is introduced in Section 2, and the system model is presented in Section 3. The dynamic game equilibrium analysis is conducted in Section 4, and the offloading algorithm is introduced in Section 5. The performance of the proposed algorithm is evaluated in Section 6. Conclusions are given in Section 7.

II. Related Work

The swift progress in artificial intelligence (AI) technology has markedly enhanced daily life with cutting-edge applications like self-driving vehicles and live traffic surveillance [19], [20]. As a technology that demands high volumes of data and processing speeds, AI requires extensive training with representative data to enhance its accuracy and reliability [21]. In reality, the computational resources available in vehicles are frequently inadequate to fulfill the high-quality service demands of AI tasks [22]. Therefore, VEC has emerged as a new computational paradigm. This technology offloads compute-intensive tasks to edge servers closer to the vehicle, reducing the impact of data transmission [23]. Compared to traditional centralized servers, edge servers are positioned closer to vehicles [24], further reducing communication-induced latency. Research in the field of VEC is dedicated to proposing optimization strategies aimed at achieving optimal resource allocation to maximize energy efficiency or minimize execution latency [25]. These resources include computational and communication resources of the servers [26].

For example, Bozorgchenani et al. [27] proposed an online learning algorithm based on the multi-armed bandit theory and an off-policy learning algorithm to minimize the task loss caused by vehicle mobility. Ren et al. [28] decomposed the task into multiple independent subtasks. Different subtasks have partially identical input data. All related subtasks can be offloaded to the same edge node by uploading the data only once. Li et al. [29] introduced an Internet of Vehicles fog-edge computing paradigm and formulated the computing task offloading as a multi-stage Stackelberg game to handle it, aiming to optimize the energy consumption of vehicles. Zhang et al. [30] minimized the offloading cost by effectively integrating service matching mining and intelligent offloading scheduling in the digital twin and physical network. A generalized FRL method has been proposed based on meta-learning techniques to achieve lower task processing latency, as referenced in [31]. References [32] and [33] analyzed the interactions between the requesting vehicles and VEC servers through Stackelberg game and found the optimal strategies for them.

However, these studies primarily focus on optimizing overall network performance, neglecting strategic behavior induced by user offloading preferences, leading to fairness issues in IoT devices and rationality problems in edge server resource utilization. To address this issue, numerous studies have been conducted. Chen et al. [34] introduced a decentralized task offloading algorithm based on game theory. In this approach, each user independently determines their offloading decision. Sun et al. [35] designed a collaborative resource allocation and task offloading algorithm, including two parts: an incentive mechanism based on a bargaining game for resource allocation and pricing within servers, aimed at promoting collaboration between tasks (vehicles) and servers; and a many-to-one matching mechanism for task offloading between servers, to stimulate edge collaboration and edge-cloud collaboration. Wang et al. [36] proposed the TM algorithm and the game-theoretic COMO algorithm based on vehicle offloading decisions to optimize resource usage.

Although the aforementioned works have significantly improved the performance of VEC networks, they do not adequately describe some vehicles' willingness to use edge server resources and the resource usage situation of edge servers, leading to persistent server overload issues. Unlike previous works, this study investigates the resource allocation and task offloading problems in VEC networks, taking into account the

latency requirements of tasks, the fairness of task offloading, vehicles' offloading willingness, and the load situation of edge servers.

III. System Model

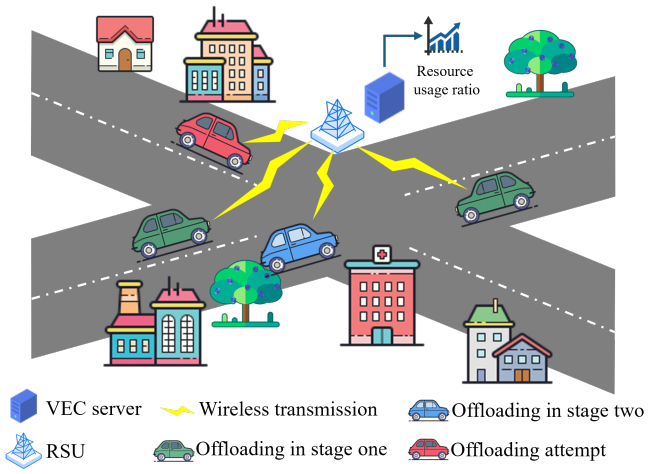


Fig. 1. System model.

As shown in Fig. 1, this system model takes into account that the edge server is executing some tasks offloaded from vehicles while new vehicles are waiting to be offloaded. In the figure, the red vehicle initiates an offloading request to the edge server. The green vehicle is in the first offloading stage. The blue vehicle is in the second offloading stage. At this time, the resource utilization rate of the edge server is constantly rising. The red vehicle needs to reasonably offload tasks while meeting the delay constraints of the tasks and avoid overloading the edge server, which may affect the execution of subsequent tasks. The purpose of the vehicles is to reduce the execution delay of their tasks through offloading, thereby enhancing their experience. The goal of the edge server is to optimize resource utilization while preventing high-load situations on the premise of meeting the needs of the vehicles. Tasks can be regarded as partially offloaded, allowing them to send part of the data to the server through the roadside unit (RSU) for processing.

The overall flow between the server and vehicles includes two stages in Fig. 2. At the beginning of offloading, vehicles initially report the tasks' relevant information to the server. During the first stage (the dynamic game offloading stage), the edge server determines a bidding function based on the task information reported by vehicles, who in turn decide how to offload according to this bidding function. If a vehicle is keen to offload and the volume of data and CPU capacity occupied meet the execution requirements of the task, the offloading proceeds. Tasks that cannot be offloaded in the dynamic game offloading stage move to the equilibrium offloading stage. In this stage, to benefit vehicles and enhance service quality, the edge server suggests the CPU capacity to be occupied and calculates the volume of data to be offloaded based on the remaining CPU capacity and the task's latency constraints, guiding vehicles in the offloading process.

The optimal decision-making strategy for vehicles involves reducing the execution delay of their tasks while considering the costs involved. The edge server sets an appropriate CPU capacity unit price to maximize its revenue.

A. Payment Model

This paper introduces a more innovative pricing scheme to address the limitations of traditional approaches. It assumes that once an edge server allocates a certain CPU capacity for a task, that task will occupy this CPU capacity until its completion.

$$f\left(C_o\right)=\gamma\frac{C_o}{C_r}+\delta \tag{1}$$

where C_o represents the CPU capacity occupied by the vehicle, and C_r denotes the remaining CPU capacity of the edge server ($C_r > 0$), tasks can only be offloaded when the edge server's remaining CPU capacity is greater than 0. γ and δ are adjustment coefficients, greater than 0, determined by the edge server.

The bidding function $f(C_o)$ is a function related to the edge server's available CPU capacity that is unused and the CPU capacity occupied by the vehicle. Since the available CPU capacity of the edge server is already determined when a new vehicle wishes to offload, the bidding function increases proportionally to the CPU capacity used by the vehicle.

The unit price is related to the edge server's remaining CPU capacity and the CPU capacity occupied by the vehicle. It is the product of the ratio of the CPU capacity occupied by the vehicle to the edge server's available CPU capacity and the bidding function.

$$p_{unit}=\frac{C_o}{C_r}f(C_o) \tag{2}$$

The tasks focused on in this paper are partially offloadable, meaning that part of the data can be offloaded. The total amount of data is D (in bits), and the quantity of data transferred to the server is d (in bits, $d \geq 0$). The amount of CPU cycles required to process 1 bit of data is represented by e. Therefore, the processing time of the task on the server can be represented as:

$$t_{edge}=\frac{de}{C_o} \tag{3}$$

The vehicle's payment cost is expressed as follows:

$$P=\frac{dep_{unit}}{C_o} \tag{4}$$

Under this payment model, the greater the amount of data offloaded and the higher the CPU capacity occupied by the vehicle, the higher the cost to the vehicle, which is also related to the edge server's available CPU capacity. When the volume of data offloaded by the vehicle remains constant, if the server has a higher remaining CPU capacity, the cost to the vehicle increases slowly when the vehicle occupies the edge server's available CPU capacity. However, if the edge server is under high load (i.e., the edge server has less remaining

CPU capacity), changes in the CPU capacity occupied by the vehicle result in significant changes in the vehicle's cost. For example, if the amount of data offloaded by the vehicle, d, is 1×10^6 bits, and the parameter, e, is 1000, then the vehicle's cost will vary depending on the available CPU capacity of the edge server. Assuming the adjustment coefficients γ and δ are 100 and 10, respectively, and the vehicle uses 1×10^9 CPU cycles per second. When the edge server's remaining CPU capacity is 100×10^9 cycles per second, the vehicle's cost is 0.0011. When the edge server's remaining CPU capacity is 10×10^9 cycles per second, the vehicle's cost is 0.2. Therefore, vehicles should pay more when the server is under high load.

B. Dynamic Game Offloading Stage

In this section, the edge server and vehicles are modeled as participants in a Stackelberg game. The primary objective of the dynamic game offloading stage is to rationally plan the resources of the server to maintain lower delays for tasks.

When a new task arrives, the vehicle reports the CPU capacity of the vehicle C_{loc}, the total amount of data D, the vehicle's payment weight η, the task's delay constraint τ, and the parameter e to the edge server, which then calculates its remaining CPU capacity C_r. Based on these data, the edge server calculates the adjustment coefficients γ and δ and provides the bidding function. At this point, the vehicle can decide how much data to offload to the edge server and how much CPU capacity to occupy. If the vehicle incurs a smaller cost, it means occupying less CPU capacity of the edge server and offloading less data, resulting in a higher execution delay for the task. Conversely, when a vehicle desires a lower execution delay and is willing to pay a higher cost, it indicates the use of more resources. Vehicles should take into account both the task's execution delay and the cost incurred, finding a balance between the two.

The task's execution delay can be represented as follows:

$$\begin{aligned} t_{exe} &= \max(t_{off}, t_{loc}) \\ t_{off} &= t_{up} + t_{edge} + t_{down} \end{aligned} \tag{5}$$

where t_{exe} represents the task's execution delay, t_{off} denotes the total time for task offloading and processing on the edge server, t_{loc} indicates the task's local processing time, t_{up} is the task's upload time, and t_{down} is the task's download time.

The upload and download times of a task are influenced by various factors, but this method primarily focuses on the transmission rate, assuming ideal conditions during upload and download. t_{up} and t_{down} can be represented as follows:

$$\begin{aligned} t_{up} &= \frac{d}{r_{up}} \\ t_{down} &= \frac{d_{io} d}{r_{down}} \end{aligned} \tag{6}$$

where r_{up} denotes the upload rate, r_{down} denotes the download rate, and d_{io} represents the ratio of the amount of data before the task execution to the amount of data after the task execution.

After the task is offloaded, the vehicle processes the remaining amount of data locally, which is $D - d$. The task's processing time on the vehicle is expressed as:

$$t_{loc} = \frac{(D-d)\, e}{C_{loc}} \tag{7}$$

According to equations (3), (5), (6), and (7), the execution delay of the task can be further expressed as follows.

$$t_{exe} = max\left(\frac{d}{r_{up}} + \frac{de}{C_o} + \frac{d_{io} d}{r_{down}}, \frac{(D-d)\, e}{C_{loc}}\right) \tag{8}$$

The cost function of the vehicle may be expressed as follows.

$$M\left(C_o, d\right) = t_{exe} + \eta P \tag{9}$$

where η represents the vehicle's payment weight, which is always positive. When the payment weight η is low, it indicates that the vehicle is more concerned about the current task's execution delay and does not mind incurring a higher cost. Conversely, when the payment weight η is high, the vehicle is less inclined to offload.

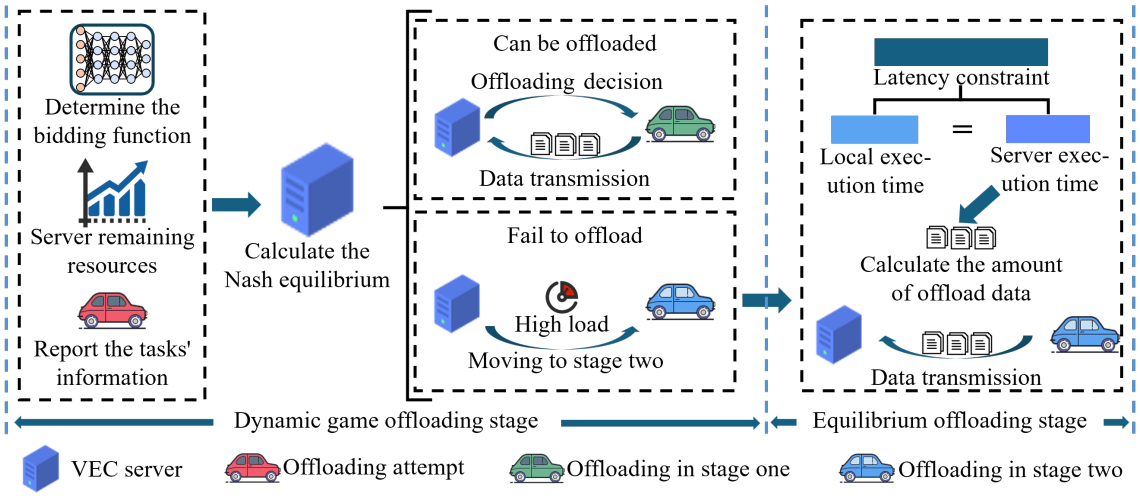


Fig. 2. The overall flow between the server and vehicles.

After the vehicle offloads, the server earns revenue. The server's objective is to maximize its revenue. Once the processing of the offloaded task is finished, the edge server will receive income. Assuming the offloaded data volume is d^{opt} and the resources of the edge server occupied are C_o^{opt}, the utility function of the server is expressed as:

$$E\left(\gamma,\delta\right)=\frac{e\gamma d^{opt}C_o^{opt}}{C_r^2}+\frac{e\delta d^{opt}}{C_r} \tag{10}$$

The edge server acts as the leader, aiming to maximize its utility function, while the vehicle, serving as the follower, seeks to minimize the cost function. On this basis, the Stackelberg dynamic game offloading stage is formulated. The offloading objective of this stage is to refine the use of edge server resources while ensuring delay constraints are met, and to avoid situations where the edge server experiences high load.

C. Equilibrium Offloading Stage

During the dynamic game offloading stage, the edge server's bidding function is inversely proportional to its remaining CPU capacity. This setting considers the real-time processing capability of the edge server and avoids high-load situations but also introduces certain issues. In the actual offloading process, when the edge server's remaining CPU capacity is scarce, the bidding function's value can become very large, leading to a correspondingly high unit price calculated using this bidding function. Although the edge server's remaining CPU capacity may be sufficient to complete the task, the high cost of offloading may deter vehicles from proceeding with the offload, thus reducing service quality. For example, if the vehicle's offloaded data volume d is 1×10^3 bits, the parameter e is 1000, the vehicle uses 1×10^6 CPU cycles per second, and the adjustment coefficients γ and δ are respectively 100 and 10. When the edge server's remaining CPU capacity is 1×10^6 cycles per second, the edge server's computing power can meet the task execution requirements, but the vehicle's cost of payment is 110.0. Therefore, this paper introduces the equilibrium offloading stage. The main goal of this stage is to address the issue of certain tasks being unable to offload due to the constraints of the bidding function and Nash equilibrium during the dynamic game offloading stage, reducing the failure rate of tasks.

In the equilibrium offloading stage, this paper proposes that the amount of data offloaded needs to ensure that the task's local processing time is equal to the total time of task processing after offloading to the edge server. So, the offloaded data volume d can be expressed in terms of the CPU capacity C_o occupied by the vehicle, as shown in equation (11). At this point, the offloaded data volume d is referred to as the equilibrium data volume $d_m(C_o)$.

$$d_m(C_o)=\frac{DeC_o}{eC_{loc}+AC_oC_{loc}} \tag{11}$$

where A represents $1/r_{up}+d_{io}/r_{down}+e/C_{loc}$.

According to the definition of task offloading delay t_{exe} from equation (8), this paper considers the longer duration between t_{loc} and t_{off} as the task's offloading delay. When the amount of data offloaded equals the equilibrium data volume, the local processing time t_{loc} and the total offloading time t_{off} are equal, as indicated by equation (12).

$$\begin{aligned} t_{exe}&=t_{off}\\ &=t_{loc}\\ &=\frac{\left(D-d_m(C_o)\right)e}{C_{loc}} \end{aligned} \tag{12}$$

The delay, t_{exe}, should be less than or equal to the task's delay constraint, τ.

$$\frac{\left(D-d_m\left(C_o\right)\right)e}{C_{loc}}\leq\tau \tag{13}$$

i.e.

$$C_o\geq\frac{C_{loc}e\left(De-C_{loc}\tau\right)}{De^2-AC_{loc}\left(De-C_{loc}\tau\right)} \tag{14}$$

In the equilibrium offloading stage, the edge server may already be under a high load. Therefore, the CPU capacity C_o occupied by the vehicle takes the minimum value, which is

$$C_o^{sec}=max\left(0,\frac{C_{loc}e\left(De-C_{loc}\tau\right)}{De^2-AC_{loc}\left(De-C_{loc}\tau\right)}\right) \tag{15}$$

where C_o^{sec} represents the CPU capacity occupied by the vehicle during the equilibrium offloading stage.

If $C_o^{sec}\leq C_r$, the task is offloaded to the edge server. The equilibrium data volume $d_m(C_o)$ can be determined by substituting the CPU capacity C_o^{sec} occupied by the vehicle during the equilibrium offloading stage into equation (11). The values of the vehicle's cost function $M(C_o,d)$ and the edge server's utility function $E(\gamma,\delta)$ can then be calculated using equations (9) and (10), respectively (with the values of γ and δ already determined during the dynamic game offloading stage).

IV. Dynamic Game Equilibrium Analysis

This section analyzes the equilibrium strategy during the dynamic game offloading stage and solves for the Nash equilibrium point.

The ideal offloading approach for vehicles is to minimize the cost function $M(C_o,d)$ by offloading an appropriate amount of data d and occupying an appropriate CPU capacity C_o, while satisfying relevant offloading constraints. The optimization problem for the vehicle's offloading decision is formulated as follows.

$$\begin{aligned} &\min_{C_o,d}\max\left(\frac{d}{r_{up}}+\frac{de}{C_o}+\frac{d_{io}d}{r_{down}},\frac{\left(D-d\right)e}{C_{loc}}\right)+\eta P\\ &\mathbf{s.t.}\\ &\quad 0\leq C_o\leq C_r\\ &\quad t_{exe}\leq t_{\mathrm{req}}\\ &\quad 0<\gamma,\quad 0<\delta,\quad 0<d_{io},\quad 0<\eta. \end{aligned} \tag{16}$$

In equation (16), to ensure that a task can be offloaded, the task's execution delay t_{exe} must be less than or equal

to the task's delay constraint τ; offloading that does not meet this condition is considered ineffective and can impact service quality. Secondly, the CPU capacity C_o occupied by the vehicle should be less than the C_r, which is determined by the objective environment. Finally, the adjustment coefficients γ and δ, the d_{io}, and the offloading weight η are all numbers greater than 0, determined by the constructed system model.

Based on the equilibrium data volume $d_m(C_o)$, the t_{exe} can be represented as follows.

$$\begin{aligned} t_{exe} =& max(t_{off}, t_{loc}) \\ =& \begin{cases} \dfrac{d}{r_{up}} + \dfrac{de}{C_o} + \dfrac{d_{io}d}{r_{down}}, & \text{if } d > d_m(C_o) \\ \dfrac{(D-d)e}{C_{loc}}, & \text{otherwise.} \end{cases} \end{aligned} \tag{17}$$

The partial derivative of the vehicle's cost function $M(C_o, d)$ with respect to the data volume d offloaded can be represented as follows:

$$\begin{aligned} &\frac{\partial M(C_o, d)}{\partial d} \\ &= \begin{cases} \frac{1}{r_{up}} + \frac{e}{C_o} + \frac{d_{io}}{r_{down}} + \frac{\eta e \gamma C_o}{C_r^2} + \frac{\eta e \delta}{C_r}, \text{if} d > d_m(C_o) \\ \frac{\eta e \gamma C_o}{C_r^2} + \frac{\eta e \delta}{C_r} - \frac{e}{C_{loc}}, \text{otherwise.} \end{cases} \end{aligned} \tag{18}$$

Let $S(C_o)$ denote the partial derivative of $\eta e \gamma C_o / C_r^2 + \eta e \delta / C_r$ with respect to the data volume d offloaded to the edge server. The shape of the vehicle's cost function $M(C_o, d)$ will change according to the variations in $S(C_o)$. A detailed analysis follows.

$$S\left(C_o\right) = \frac{\eta e \gamma C_o}{C_r^2} + \frac{\eta e \delta}{C_r} \tag{19}$$

If $S(C_o) > e/C_{loc}$, $\forall d \in [0, D]$ and $\forall C_o \in \{C_o | S(C_o) > e/C_{loc}\}$, $\partial M(C_o, d)/\partial d > 0$, meaning the vehicle's cost function $M(C_o, d)$ is a function that increases monotonically with respect to the data volume d offloaded to the edge server. In this scenario, vehicles are unlikely to benefit regardless of the amount of data they offload.

If $S(C_o) < e/C_{loc}$, $\forall d \in [0, d_m(C_o))$ and $\forall C_o \in \{C_o | S(C_o) < e/C_{loc}\}$, $\partial M(C_o, d)/\partial d < 0$, indicating that the vehicle's cost function $M(C_o, d)$ is a monotonically decreasing function concerning the data volume d offloaded to the edge server. $\forall d \in (d_m(C_o), D]$ and $\forall C_o \in \{C_o | S(C_o) < e/C_{loc}\}$, $\partial M(C_o, d)/\partial d > 0$, meaning the vehicle's cost function $M(C_o, d)$ becomes a monotonically increasing function beyond this point. When $d = d_m(C_o)$, the vehicle's cost function $M(C_o, d)$ has a minimum value, which can be expressed as follows:

$$\begin{aligned} &M\Big(C_o, d_m(C_o)\Big) = \\ &\frac{De}{C_{loc}}\left[1 + \frac{eC_o}{C_r^2 C_{loc}}\left(\frac{\eta\gamma C_{loc} C_o + \eta\delta C_r C_{loc} - C_r^2}{e + AC_o}\right)\right] \end{aligned} \tag{20}$$

If $S(C_o) = e/C_{loc}$, $\forall d \in [0, d_m(C_o)]$ and $\forall C_o \in \{C_o | S(C_o) = e/C_{loc}\}$, $\partial M(C_o, d)/\partial d = 0$, meaning the vehicle's cost function $M(C_o, d)$ is constant. In this case, even if vehicles do not need to offload more data, they would opt to offload the equilibrium data volume $d_m(C_o)$. $\forall d \in (d_m(C_o), D]$ and $\forall C_o \in \{C_o | S(C_o) = e/C_{loc}\}$, $\partial M(C_o, d)/\partial d > 0$, indicating that the vehicle's cost function $M(C_o, d)$ is a function that increases monotonically concerning the data volume d offloaded to the edge server.

When $S(C_o) < e/C_{loc}$, the vehicle's cost function $M(C_o, d)$ has a minimum value, and vehicles can benefit from offloading, that is, offloading the equilibrium data volume $d_m(C_o)$. When $S(C_o) = e/C_{loc}$, $\forall d \in [0, d_m(C_o)]$, vehicles may also choose to offload the equilibrium data volume $d_m(C_o)$. However, when $S(C_o) > e/C_{loc}$, the vehicle's cost function $M(C_o, d)$ is a function that increases monotonically concerning the data volume d offloaded to the edge server, and vehicles cannot benefit from offloading. Therefore, only when $S(C_o) \leq e/C_{loc}$ can vehicles possibly proceed with offloading. Based on the definition of $S(C_o)$, the upper limit of the CPU capacity C_o occupied by the vehicle can be expressed as follows:

$$C_o \leq \frac{C_r}{\gamma}\left(\frac{C_r}{\eta C_{loc}} - \delta\right) \tag{21}$$

Therefore, the vehicle's optimal policy can be expressed by the following equation.

$$d^*(C_o) = \begin{cases} d_m(C_o), & \text{if} \quad C_o \leq \dfrac{C_r}{\gamma}\Big(\dfrac{C_r}{\eta C_{loc}} - \delta\Big) \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

If the vehicle decides to offload, then the amount of data offloaded is the equilibrium data volume $d_m(C_o)$, and the vehicle's cost function $M(C_o, d)$ equals the value given by equation (20). At this point, the t_{loc} and the t_{off} are equal. The t_{exe} should be less than or equal to the τ. By combining equations (14) and (21), the range of values for the CPU capacity C_o occupied by the vehicle can be determined as follows:

$$\begin{aligned} C_o &\geq \frac{C_{loc} e \left(De - C_{loc}\tau\right)}{De^2 - AC_{loc}\left(De - C_{loc}\tau\right)} \\ C_o &\leq \frac{C_r}{\gamma}\left(\frac{C_r}{\eta C_{loc}} - \delta\right) \end{aligned} \tag{23}$$

Based on equations (16), (20), (22), and (23), the vehicle's uninstallation optimization problem is further formulated as follows:

$$\begin{aligned} &\min_{C_o} \frac{De}{C_{loc}}\left[1 + \frac{eC_o}{C_r^2 C_{loc}}\left(\frac{\eta\gamma C_{loc} C_o + \eta\delta C_r C_{loc} - C_r^2}{e + AC_o}\right)\right] \\ &\mathbf{s.t.} 0 \leq d \leq D \\ &\quad 0 < \gamma, 0 < \delta, 0 < d_{io}, 0 < \eta \\ &\quad C_o \geq \frac{C_{loc} e \left(De - C_{loc}\tau\right)}{De^2 - AC_{loc}\left(De - C_{loc}\tau\right)} \\ &\quad C_o \leq \frac{C_r}{\gamma}\left(\frac{C_r}{\eta C_{loc}} - \delta\right) \end{aligned} \tag{24}$$

In equation (24), the d is determined, but the amount of CPU capacity C_o that the vehicle needs to occupy on the edge

server is not specified. The following analysis will determine how much CPU capacity C_o the vehicle needs to occupy on the edge server.

Taking the second derivative of equation (24) yields:

$$\frac{d^2M(C_o)}{dC_o^2} = \frac{2De^3}{C_{loc}^2C_r^2}\left\{\frac{\eta\gamma eC_{loc}+A(C_r^2-\eta\delta C_{loc}C_r)}{(e+AC_o)^3}\right\} \tag{25}$$

Given the constraint from equation (21), $C_r^2-\eta\delta C_{loc}C_r \geq \gamma\eta C_{loc}C_o$, and since $\gamma\eta C_{loc}C_o$ is always greater than or equal to 0, it follows that $0 \leq d^2M(C_o)/dC_o^2$. This implies that $M(C_o)$ is a convex function. Therefore, there exists a solution that minimizes the cost function $M(C_o)$.

By setting the first derivative of the cost function $M(C_o)$ equal to zero, the solution that minimizes $M(C_o)$ can be determined.

$$C_o' = \frac{-\eta\gamma e \pm \sqrt{\eta^2\gamma^2e^2-\eta\gamma eA\left(\eta C_r\delta-\frac{C_r^2}{C_{loc}}\right)}}{\eta\gamma A} \tag{26}$$

Based on the constraint from equation (21), $\eta C_r\delta - C_r^2/C_{loc}$ is always less than 0, and the CPU capacity C_o occupied on the edge server is greater than or equal to 0. Therefore, the optimal CPU capacity C_o occupied on the edge server is a positive value, which implies that

$$C_o^* = \frac{-\eta\gamma e + \sqrt{\eta^2\gamma^2e^2-\eta\gamma eA\left(\eta C_r\delta-\frac{C_r^2}{C_{loc}}\right)}}{\eta\gamma A} \tag{27}$$

where C_o^* signifies the attainable Nash equilibrium of occupied CPU capacity on edge servers.

Consequently, it can be concluded that during the dynamic game offloading stage, the optimal offloading scheme can be articulated as follows:

$$C_o^{opt} = \begin{cases} C_o^* & \text{if} \quad \eta C_r\delta - \frac{C_r^2}{C_{loc}} < 0, \quad C_{min} \leq C_o^* \leq C_{max} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

$$d^{opt} = d_m\left(C_o^{opt}\right) = \frac{DeC_o^{opt}}{C_{loc}\left(e+AC_o^{opt}\right)} \tag{29}$$

$$C_{min} = max\left(0, \frac{C_{loc}e\left(De-C_{loc}\tau\right)}{De^2-AC_{loc}\left(De-C_{loc}\tau\right)}\right) \tag{30}$$

$$C_{max} = min\left(C_r, \frac{C_r}{\gamma}\left(\frac{C_r}{\eta C_{loc}}-\delta\right)\right) \tag{31}$$

Herein, C_o^{opt} represents the CPU capacity for optimal offloading, d^{opt} denotes the data volume for optimal offloading, C_{min} signifies the lower limit of occupied CPU capacity on edge servers, and C_{max} illustrates the upper limit of occupied CPU capacity on edge servers.

Algorithm 1 Task Offloading Algorithm.

```
Initialize C_o^opt = 0 and C_o^sec = 0 .
while a new task arrives do
    vehicle reports(C_loc, D, η, τ, e).
    Get remaining CPU capacity C_r of the edge server.
    Compute adjustment coefficients γ and δ.
    if ηC_rδ − C_r^2/C_loc < 0 then
        Calculate C_o^* using formula (27).
        Calculate C_min using formula (30).
        if C_r/(ηC_loc) − δ > 0 then
            Calculate C_max using formula (31).
            if C_min ≤ C_o^* ≤ C_max then
                C_o^opt = C_o^*
                Calculate d^opt using formula (29).
            end if
        end if
    end if
    if C_o^opt == 0 then
        Calculate C_o^sec using formula (15).
        if C_o^sec ≤ C_r then
            Calculate d_m(C_o^sec) by (11).
        end if
    end if
    Calculate M(C_o, d) by (9).
    Calculate E(γ, δ) by (10).
end while
```

V. Task Offloading Algorithm

This section first introduces the algorithm for task offloading, followed by an explanation of how edge servers determine the adjustment coefficients γ and δ for the bidding function $f(C_o)$.

A. Algorithmic Process

The overall algorithm flow is depicted in Algorithm 1. It begins by initializing the CPU capacity for optimal offloading C_o^{opt} and the CPU capacity occupied by vehicles during the equilibrium offloading stage C_o^{sec}. Upon the arrival of new tasks at the edge server, the system enters the dynamic game offloading stage (steps 3-16 of the algorithm). Vehicles report to the edge server the local CPU capacity C_{loc}, the total data volume D, the payment weight of the vehicle η, the τ, and the parameter e, while the edge server calculates its remaining CPU capacity C_r. Based on this information, the edge server calculates the adjustment coefficients γ and δ. If the condition $\eta C_r\delta - C_r^2/C_{loc} < 0$ holds, it indicates the existence of an optimal offloading scheme. The CPU capacity occupied at the edge server to achieve Nash equilibrium C_o^* and the lower limit of the CPU capacity occupied at the edge server C_{min} are calculated using formulas (27) and (30), respectively. When $C_r/\eta C_{loc} - \delta > 0$, the upper limit of the CPU capacity occupied at the edge server C_{max} is calculated using formula (31). Task offloading can proceed if the Nash equilibrium of occupied CPU capacity at the edge server C_o^* falls within the range of C_{min} and C_{max}. Let $C_o^{opt} = C_o^*$. If $C_o^{opt} = 0$, it indicates that the task could not be offloaded during the

dynamic game offloading stage due to load issues at the edge server or other problems. Tasks that cannot be offloaded enter the equilibrium offloading stage (steps 17-22 of the algorithm). Firstly, calculate the CPU capacity occupied by vehicles during the equilibrium offloading stage C_o^{sec} using the formula (15). Next, if $C_o^{sec} \leq C_r$, it implies that the current resources of the server are sufficient to execute the task, allowing for its offloading. Lastly, sequentially calculate the volume of data offloaded, the cost function, and the utility function (steps 23-24 of the algorithm).

B. The Adjustment Coefficients

In this paper, the adjustment coefficients γ and δ for the bidding function are determined by the edge server. The optimal offloading decision of vehicles changes with variations in the adjustment coefficients γ and δ. During the dynamic game offloading stage, unreasonable values of γ and δ may lead to irrational vehicle cost functions, dampening vehicles' enthusiasm for offloading, reducing the utility of the edge server, and consequently transitioning to the equilibrium offloading stage.

The main challenge in addressing this issue lies in the difficulty of determining a fixed solution, as the adjustment coefficients γ and δ upon which the vehicle's offloading decision depends are constantly changing in a complex environment. To tackle this challenge, a heuristic search algorithm can be employed to explore possible solutions. However, the efficiency of heuristic searches is greatly affected by the settings of search intervals and ranges, and the execution time of the algorithm can significantly increase. If the edge server stores many historical search results, a regression function can be constructed to approximate these solutions. In this manner, the server can quickly provide recommendations based on the bidding mechanism when a vehicle requests offloading. Consequently, this study proposes a supervised learning-based approach to determine the adjustment coefficients γ and δ by analyzing accumulated data, thereby finding effective offloading strategies.

This method employs a feedforward neural network with regression based on supervised learning. The network structure comprises three hidden layers, with the first layer consisting of 512 hidden neurons, the second layer of 256 hidden neurons, and the third layer of 128 hidden neurons. All layers' activation function is the ReLU function. The paper generated one million data points for training and three hundred thousand for testing. Each data point includes the C_{loc}, D, η, τ, e, and the C_r. After training, the edge server obtains a model for calculating the adjustment coefficients γ and δ. When the system is running, it can quickly provide reasonable adjustment coefficients γ and δ by loading the trained model.

VI. Experimental Evaluation

A. Experimental Setup

In the simulation experiments, it was assumed that the size ratio of data before and after processing, d_{io}, is 0.2, and the total data volume, D, is uniformly distributed within the range of [100, 300] Kbytes. The local CPU capacities, C_{loc}, of vehicles were based on the Samsung Galaxy S10 (2.84×10^9 cycles/second), Apple iPhone X (2.39×10^9 cycles/second), and Google Pixel 3 (2.5×10^9 cycles/second). The edge server's CPU capacity was referenced to the AMD EPYC 7002 server's capacity [37], which is 110×10^9 cycles/second. The parameter, e, was set to 2640, based on the computational intensity of a video game with 400 frames [38]. The latency constraint, τ, was randomly chosen from the collection {100ms, 500ms, 1000ms}. The upload and download rates were set based on the mean 5G upload and download rates of the world's top 10 telecom firms, at 41.2 and 360.3 Mb/s, respectively. The task arrival rates, R, were set to 1, 2, and 3, corresponding to 1, 2, and 3 tasks arriving every 100ms, respectively. The experiment ran for 300 episodes, with 100 episodes each for task arrival rates R of 1, 2, and 3.

The simulation experiments were compared against the following baselines: executing all tasks locally, deterministic pricing model, uniform pricing model [39], and differential pricing model [40]. The deterministic pricing model and uniform pricing model are classical algorithms used for task offloading. The differential pricing model is a more novel algorithm. These three algorithms perform well under different task offloading scenarios. Therefore, these algorithms were selected for comparative experiments. The experiments compared several performance metrics under different task arrival rates, including the average cost to vehicles, average execution latency, revenue for the edge server, task failure rate, and resource utilization rate of the edge server.

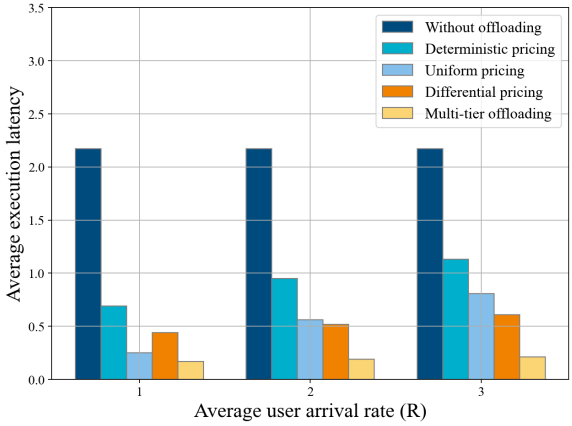


Fig. 3. Average execution latency.

B. Average Execution Latency

Fig. 3 presents the experimental results of the average execution latency for different models. The execution latencies for the four pricing models are calculated using equation (8).

In Fig. 3, across various task arrival rates, the method of executing all tasks locally maintains an average latency of 2.17 seconds. Compared to this, the average execution latencies of the other three pricing models and the multi-tier offloading model proposed in this paper are significantly lower. The deterministic pricing model has an average execution latency of 0.69 seconds at a task arrival rate of 1; however, its latency

significantly increases with the task arrival rate, reaching 1.13 seconds at a task arrival rate of 3. The uniform pricing model maintains a low average execution latency of 0.25 seconds at a task arrival rate of 1, outperforming the differential pricing model, but it fails to keep a low average latency as the task arrival rate increases, with latencies of 0.56 seconds at a rate of 2 and 0.81 seconds at a rate of 3, significantly higher than those of the differential pricing model. While the differential pricing model does not perform as well at a task arrival rate of 1, with an average latency of 0.44 seconds, its latency only slightly increases as the number of tasks grows, with latencies of 0.52 seconds at a rate of 2 and 0.61 seconds at a rate of 3. The multi-tier offloading model proposed in this paper consistently maintains the lowest average execution latency, regardless of the task arrival rate, with corresponding latencies of 0.17 seconds, 0.19 seconds, and 0.21 seconds for rates 1, 2, and 3, respectively.

At a task arrival rate of 1, fewer tasks are offloaded, which does not become highly loaded, and CPU capacity is relatively abundant. Thus, the uniform and differential pricing models perform well. However, at a task arrival rate of 3, the edge server experiences high load, with limited remaining CPU capacity, and the CPU capacity for task execution cannot always be satisfied, leading to a significant increase in average execution latency for both the uniform and differential pricing models. The multi-tier offloading model considers the C_r and uses the edge server's CPU capacity more rationally than the uniform and differential pricing models. Despite high task arrival rates, the multi-tier offloading model finely utilizes CPU capacity through the bidding function. If some tasks fail to offload in the first stage, they can be offloaded in the second stage, maintaining a low execution latency.

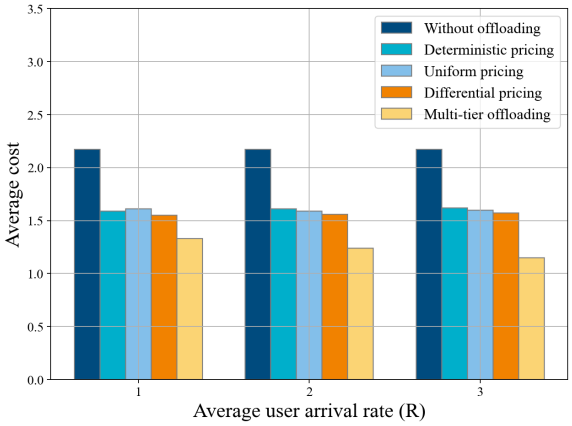


Fig. 4. Average cost.

C. Average Cost

Fig. 4 displays the experimental results for the average cost to vehicles across different models. The average cost to vehicles for the four pricing models is determined by the weighted aggregate of the delay and the vehicle's payment expense.

In Fig. 4, across various task arrival rates, the method of executing all tasks locally has an average cost of 2.17. Compared to the other three pricing models, the average cost of the multi-tier offloading model is significantly lower. The average costs to vehicles for the other three pricing models are nearly identical. However, due to the lower average execution latency of tasks, the multi-tier offloading model results in a lower average cost to vehicles. As the rate of task arrival increases, the average cost to vehicles paradoxically decreases, indicating that higher task arrival rates lead to less remaining CPU capacity on the edge server and more tasks entering the equilibrium offloading stage, thereby reducing the average cost to vehicles.

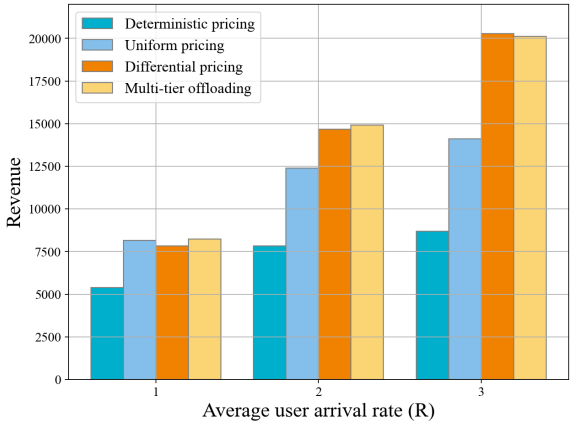


Fig. 5. Revenue for the edge server.

D. Revenue for the Edge Server

Fig. 5 presents the experimental results for the revenue of the edge server across different models. At a task arrival rate of 1, the revenue for the deterministic pricing model is 5401.56, for the uniform pricing model it is 8169.25, for the differential pricing model it is 7847.91, and for the multi-tier offloading model it is 8234.64. As the rate of task arrival increases, so does the revenue for the edge server. However, at a task arrival rate of 3, the revenue for the deterministic pricing model reaches 8698.62, for the uniform pricing model it is 14099.19, for the differential pricing model it skyrockets to 20267.96, and for the multi-tier offloading model it slightly trails at 20119.2. The revenue from the multi-tier offloading model is slightly lower than that of the differential pricing model. The differential pricing model, which occupies more of the edge server's CPU resources, charges a higher price per unit. When the edge server is under a high load, the differential pricing model does not optimize for this scenario; even with a high load, as long as resources can satisfy the task execution, the model will still offload tasks. However, the multi-tier offloading model, under high edge server load conditions, charges a much higher price per unit than the differential pricing model. Vehicles in the dynamic game offloading stage, due to the high price per unit, will not offload tasks to the edge server. When tasks are offloaded during the equilibrium offloading stage, vehicles follow the edge server's guidance to use the minimum CPU capacity that can meet the latency constraint for offloading. Therefore, the revenue

from the multi-tier offloading model is lower than that of the differential pricing model.

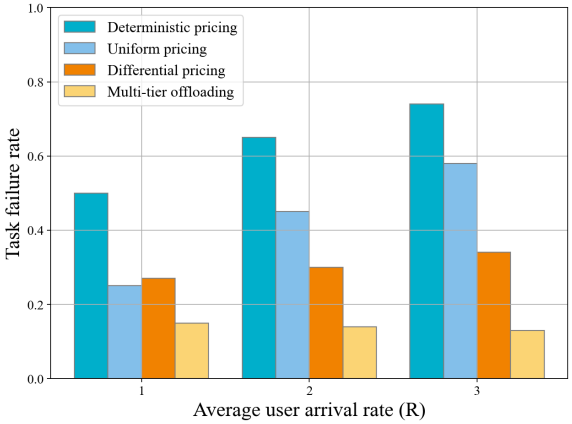


Fig. 6. Task failure rate.

E. Task Failure Rate

Fig. 6 displays the experimental results for the task failure rate across different models. A failed task is defined as one whose execution time exceeds its latency constraint. The task failure rates for the deterministic pricing model are 0.5, 0.65, and 0.74; for the uniform pricing model, they are 0.25, 0.45, and 0.58; for the differential pricing model, they are 0.27, 0.3, and 0.34; and for the multi-tier offloading model, they are 0.15, 0.14, and 0.13, respectively. For the deterministic, uniform, and differential pricing models, the task failure rate increases as the task arrival rate increases. Conversely, the multi-tier offloading model shows a decrease in task failure rate as the rate of task arrival increases. This outcome can be attributed to two main reasons:

1) High Task Arrival Rates and Edge Server Load Consideration: During high task arrival rates, the dynamic game offloading stage thoroughly considers the high load issue of the edge server. When the edge server is under high load, tasks cannot execute the optimal offloading strategy due to high prices, and if the optimal offloading strategy were executed, the C_r could not satisfy the execution of subsequent tasks in a short time. Hence, despite the high load, the edge server's remaining CPU capacity can still meet the task execution requirements, but due to the bidding function's limitations, tasks cannot be optimally offloaded.
2) Offloading in the Equilibrium Stage: Tasks that do not execute the optimal offloading strategy during the dynamic game stage proceed with offloading during the equilibrium stage. When tasks are offloaded in the equilibrium stage, they tend to occupy less CPU capacity to meet their latency constraints, allowing the edge server's remaining CPU capacity to reduce the task failure rate as much as possible.

F. Edge Server Resource Utilization Rate

Fig. 7 shows the server's resource utilization rate within 10 seconds for the three models.

In Fig. 7, the first row indicates the resource utilization rate of the server under the uniform pricing model at different task arrival rates. Regardless of whether the task arrival rate is low or high, the edge server is mostly under high load. This is because, in the uniform pricing model, the vehicle's payment cost is only related to the size of the task being processed, leading vehicles to occupy a significant amount of the edge server's resources. Coupled with the task failure rate, the constant high load on the edge server prevents many tasks from obtaining sufficient resources to meet latency constraints, resulting in a higher task failure rate. This model fails to refine the use of edge server resources effectively, leading to poor service quality.

The second row shows the performance under the differential pricing model. At low task arrival rates, the edge server rarely experiences high load. However, as the task arrival rate increases, the edge server's resources gradually reach a bottleneck, and the task failure rate increases. Compared to the uniform pricing model, this model reduces the overload on the edge server but does not perform well at high task arrival rates.

The third row in Fig. 7 shows the performance under the multi-tier offloading model. At a task arrival rate of R=1, the resource utilization rates of the edge server in the differential pricing model and the multi-tier offloading model are similar. This similarity is mainly because, at a task arrival rate of R=1, the edge server's remaining CPU resources are almost equivalent to the total CPU resources, and the majority of tasks are offloaded during the dynamic game stage. The results of the multi-tier offloading model approximate those of the differential pricing model. However, there are differences, indicating that the equilibrium offloading stage ensures the smooth execution of a small portion of tasks even at low task arrival rates. From the resource utilization rate results at task arrival rates of R=2 and R=3, it is evident that the multi-tier offloading model reduces the occurrence of edge server overload, demonstrating better performance. At higher task arrival rates, with less remaining CPU capacity on the edge server and higher offloading costs, vehicles are forced to offload less data, occupying the server's fewer resources. The high prices in the dynamic game offloading stage might even prevent tasks from being optimally offloaded. The equilibrium offloading stage ensures the smooth offloading of tasks.

G. Offloading Rates of Different Task Stages

TABLE I
OFFLOADING RATES OF DIFFERENT TASK STAGES.

	R=1	R=2	R=3
The first stage	0.71	0.69	0.65
The second stage	0.14	0.17	0.22

The dynamic game offloading stage is the initial phase in the multi-tier offloading model, primarily leveraging dynamic game theory to optimize the utilization of edge server resources. In this stage, each user's offloading decision is adjusted based on the resource usage of the edge servers to

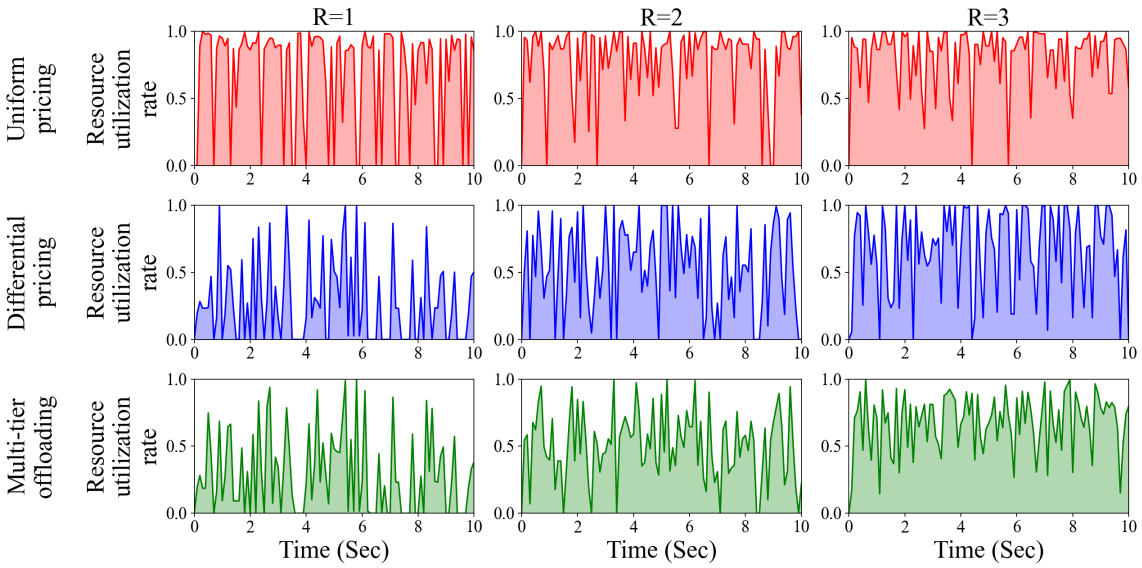


Fig. 7. Edge server resource utilization rate.

achieve optimal task processing efficiency. According to Fig. 7 and Tabel I, the task processing ratios in the dynamic game offloading stage under different user arrival rates are 0.71, 0.69, and 0.65, respectively. This indicates that the dynamic game offloading stage maintains high task processing ratios and resource utilization across various workloads.

The equilibrium offloading stage is the second phase in the multi-tier offloading model, designed to handle tasks that cannot be offloaded during the dynamic game offloading stage. In this phase, the edge server's resources can still accommodate certain tasks, allowing those that were not offloaded in the dynamic game stage to be processed. According to Fig. 7 and Tabel I, the task processing ratios in the equilibrium offloading stage under different user arrival rates are 0.14, 0.17, and 0.22, respectively. Although the processing ratios are lower, this stage effectively supplements the dynamic game offloading stage by ensuring that the overall task processing capacity of the system is maintained.

VII. Conclusion

This paper proposes a multi-tier offloading task processing scheme in the VEC environment, aiming to refine the use of edge server resources and improve service quality. The proposed multi-tier offloading model consists of two stages. The dynamic game offloading stage refines the use of edge server resources, and the second stage reduces the task failure rate to ensure service quality. This paper constructs a simulation environment based on the characteristics of video game tasks in the consumer electronics field. Experiments compare the average cost of vehicles, average execution delay, income of edge servers, task failure rate, and resource utilization rate of edge servers under different task arrival rates. The resources of edge servers are limited, and the model proposed in this paper can effectively refine the use of edge server resources, especially under high task arrival rates, effectively alleviating the overload of edge servers and significantly reducing the task failure rate.

There are several directions to extend this work. First, the work in this paper focuses only on task offloading scenarios for a single edge server, which limits its application to more complex and practical multi-edge server environments. Dynamic changes in tasks and resources tend to be more frequent and complex in multi-edge server environments. Second, algorithm performance can be evaluated in a demonstration system under which many practical problems should be solved. Finally, game theory and multi-agent reinforcement learning techniques can be combined to further understand the policy interactions among devices.

References

[1] X. Zhou, Q. Yang, Q. Liu, W. Liang, K. Wang, Z. Liu, J. Ma, and Q. Jin, "Spatial–temporal federated transfer learning with multi-sensor data fusion for cooperative positioning," *Information Fusion*, vol. 105, p. 102182, 2024.

[2] L. Zeng, Q. Liu, S. Shen, and X. Liu, "Improved double deep q network-based task scheduling algorithm in edge computing for makespan optimization," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 806–817, 2024.

[3] J. Zhang, Y. Wu, G. Min, and K. Li, "Neural network-based game theory for scalable offloading in vehicular edge computing: A transfer learning approach," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2024.

[4] C. Liu, M. Zhao, H. Wang, B. Cheng, J. Liu, and P. Yuan, "Stackelberg-game computation offloading scheme for parked vehicle-assisted vec and experiment analysis," *IEEE Transactions on Intelligent Vehicles*, pp. 1–12, 2024.

[5] X. Huang, Y. Zhang, Y. Qi, C. Huang, and M. S. Hossain, "Energy-efficient uav scheduling and probabilistic task offloading for digital twin-empowered consumer electronics industry," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2145–2154, 2024.

[6] Y. Chen, J. Zhao, J. Hu, S. Wan, and J. Huang, "Distributed task offloading and resource purchasing in noma-enabled mobile edge computing: Hierarchical game theoretical approaches," *ACM Trans. Embed. Comput. Syst.*, vol. 23, no. 1, jan 2024. [Online]. Available: https://doi.org/10.1145/3597023

[7] Q. Liu, J. Sun, Y. Zhang, and X. Liu, "Denmerd: a feature enhanced approach to radar beam blockage correction with edge-cloud computing," *Journal of Cloud Computing*, vol. 13, no. 1, p. 32, 2024.

[8] J. Tian, L. Zhu, F. R. Yu, H. Wang, and T. Tang, "Optimizing edge resources in intelligent railway construction: A two-level game approach," *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2024.

[9] C.-C. Lin, Y. Chiang, and H.-Y. Wei, "Multi-service edge computing management with multi-stage coalition game task offloading," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2024.

[10] J. Zhang, B. Zhang, and Z. Han, "Coalition formation game based information-energy collaboration in vehicle edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 7717–7727, 2023.

[11] Z. Ning, P. Dong, X. Wang, X. Hu, L. Guo, B. Hu, Y. Guo, T. Qiu, and R. Y. K. Kwok, "Mobile edge computing enabled 5g health monitoring for internet of medical things: A decentralized game theoretic approach," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 463–478, 2021.

[12] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 15 513–15 526, 2023.

[13] M. K. Mondal, S. Banerjee, D. Das, U. Ghosh, M. S. Al-Numay, and U. Biswas, "Toward energy-efficient and cost-effective task offloading in mobile edge computing for intelligent surveillance systems," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4087–4094, 2024.

[14] X. Zhu, Y. Luo, A. Liu, N. N. Xiong, M. Dong, and S. Zhang, "A deep reinforcement learning-based resource management game in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2422–2433, 2022.

[15] X. Zhou, X. Zheng, X. Cui, J. Shi, W. Liang, Z. Yan, L. T. Yang, S. Shimizu, and K. I.-K. Wang, "Digital twin enhanced federated reinforcement learning with lightweight knowledge distillation in mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3191–3211, 2023.

[16] C. Liu and K. Liu, "Toward reliable dnn-based task partitioning and offloading in vehicular edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3349–3360, 2024.

[17] L. Li, Q. Qiu, Z. Xiao, Q. Lin, J. Gu, and Z. Ming, "A two-stage hybrid multi-objective optimization evolutionary algorithm for computing offloading in sustainable edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 735–746, 2024.

[18] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, and J. J. P. C. Rodrigues, "A-dafto: Artificial cap deferred acceptance-based fair task offloading in complex iot-fog networks," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 914–926, 2023.

[19] Z. Sun, G. Sun, Y. Liu, J. Wang, and D. Cao, "Bargain-match: A game theoretical approach for resource allocation and task offloading in vehicular edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1655–1673, 2024.

[20] X. Zhou, W. Liang, J. She, Z. Yan, and K. I.-K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5308–5317, 2021.

[21] F. Zhang and M. M. Wang, "Stochastic congestion game for load balancing in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 778–790, 2021.

[22] Z. Xue, C. Liu, C. Liao, G. Han, and Z. Sheng, "Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 5, pp. 6709–6722, 2023.

[23] X. Zhou, W. Liang, K. I.-K. Wang, Z. Yan, L. T. Yang, W. Wei, J. Ma, and Q. Jin, "Decentralized p2p federated learning for privacy-preserving and resilient mobile robotic systems," *IEEE Wireless Communications*, vol. 30, no. 2, pp. 82–89, 2023.

[24] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1243–1259, 2022.

[25] X. Xu, Q. Jiang, P. Zhang, X. Cao, M. R. Khosravi, L. T. Alex, L. Qi, and W. Dou, "Game theory for distributed iov task offloading with fuzzy neural network in edge computing," *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 11, pp. 4593–4604, 2022.

[26] Q. Liu, L. Zeng, M. Bilal, H. Song, X. Liu, Y. Zhang, and X. Cao, "A multi-swarm pso approach to large-scale task scheduling in a sustainable supply chain datacenter," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 4, pp. 1667–1677, 2023.

[27] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4233–4248, 2022.

[28] H. Ren, K. Liu, C. Liu, G. Yan, and Y. Li, "An approximation algorithm for joint data uploading and task offloading in iov," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3018–3030, 2024.

[29] Y. Li, B. Yang, H. Wu, Q. Han, C. Chen, and X. Guan, "Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 969–15 982, 2022.

[30] K. Zhang, J. Cao, and Y. Zhang, "Adaptive digital twin and multi-agent deep reinforcement learning for vehicular edge computing and networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1405–1413, 2022.

[31] P. Consul, I. Budhiraja, D. Garg, N. Kumar, R. Singh, and A. S. Almogren, "A hybrid task offloading and resource allocation approach for digital twin-empowered uav-assisted mec network using federated reinforcement learning for future wireless network," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3120–3130, 2024.

[32] B. Liang, R. Fan, H. Hu, Y. Zhang, N. Zhang, and A. Anpalagan, "Nonlinear pricing based distributed offloading in multi-user mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 1077–1082, 2021.

[33] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3247–3257, 2021.

[34] Y. Chen, J. Zhao, Y. Wu, J. Huang, and X. Shen, "Qoe-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 769–784, 2024.

[35] Z. Sun, G. Sun, Y. Liu, J. Wang, and D. Cao, "Bargain-match: A game theoretical approach for resource allocation and task offloading in vehicular edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1655–1673, 2024.

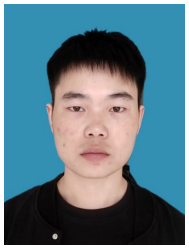
[36] H. Wang, T. Lv, Z. Lin, and J. Zeng, "Energy-delay minimization of task migration based on game theory in mec-assisted vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8175–8188, 2022.

[37] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 49–62. [Online]. Available: https://doi.org/10.1145/1814433.1814441

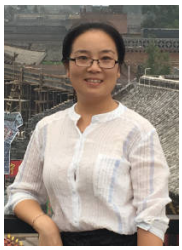
[38] K. Raaen and T.-M. Grønli, "Latency thresholds for usability in games: A survey," in *Norsk Informatikkonferanse*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:17755392

[39] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 420–423, 2018.

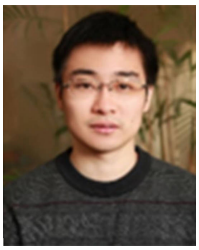
[40] H. Seo, H. Oh, J. K. Choi, and S. Park, "Differential pricing-based task offloading for delay-sensitive iot applications in mobile edge computing system," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19 116–19 131, 2022.



Haiyang Lin received the B.S. degree in Internet of Things Engineering from Nanjing University of Information Science and Technology in 2022. He is currently working toward the master's degree in electronic information with the School of Nanjing University of Information Science and Technology, Nanjing. His recent research focus is on edge computing.



Bo Xiao is with Jiangsu Province Engineering Research Center of Advanced Computing and Intelligent Services, Nanjing 210044, China, and also with the School of Chemistry and Materials Science, Nanjing University of Information Science and Technology, Nanjing 210044, China. Her recent research focuses on edge collaboration, edge intelligence.



Xiaokang Zhou (Member, IEEE) received the Ph.D. degree in human sciences from Waseda University, Japan, in 2014. He is currently an Associate Professor with the Faculty of Data Science, Shiga University, Hikone, Japan. From 2012 to 2015, he was a Research Associate with the Faculty of Human Sciences, Waseda University, Japan. He also works as a Visiting Researcher with the RIKEN Center for Advanced Intelligence Project (AIP), RIKEN, Tokyo, Japan, since 2017. He has been engaged in interdisciplinary research works in the fields of computer science and engineering, information systems, and social and human informatics. His recent research interests include ubiquitous computing, big data, machine learning, behavior and cognitive informatics, cyber-physical-social systems, and cyber intelligence and security. Dr. Zhou is a member of the IEEE Computer Society, and ACM, USA, IPSJ, and JSAI, Japan, and CCF, China.

Yonghong Zhang (Member, IEEE) received the Ph.D. degree in mechanical and power engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005. His research interests include meteorological disaster monitoring and warning, edge cloud cooperation, IoT continuum, and smart grid. His recent research interests include but are not limited to pattern recognition and intelligent systems, remote sensing big data analysis and deep learning, intelligent equipment and IoT system integration, and higher education teaching management and research work.

Xiaodong Liu received the Ph.D. degree in computer science from De Montfort University. In 1999, he joined Edinburgh Napier University, where he is a Reader and currently leading the Software Systems Research Group with IIDI. He was the Director of the Centre for Information and Software Systems. He is an Active Researcher in software engineering with internationally excellent reputation and leading expertise in contex-aware adaptive services, service evolution, mobile clouds, pervasive computing, software reuse, and green software engineering.