D2D and Edge Server-Enabled Computation Offloading for Resource-Constrained Wireless Networks

Yuan Yuan, Bin Yang, Wei Su, Hongke Zhang, Qi Liu, Tarik Taleb

***Abstract*—For the computation offloading via device-to-device (D2D) terminals and edge servers in a resource-constrained wireless network (RCWN), mobile users can choose to offload their tasks to nearby D2D terminals or edge servers according to quality of service (QoS) requirements (e.g., load balancing at the network edge) by mobile edge computing. To this end, we first formulate computation offloading as a multi-user collaborative resource dynamic management optimization problem that aims to maximize user satisfaction utility function, carefully considering critical issues like the non-uniform distribution of computational resources, user's risk awareness, and the dynamic changes between computing-intensive regions and computing-sparse regions. This is a nonlinear and nonconvex optimization problem, which is generally difficult to be solved. We then construct a resource management scheme for resource allocation of the edge server based on convex optimization. Furthermore, we propose a dynamic offloading update strategy achieving the maximum of user satisfaction utility function based on game theory. The simulation results are presented to show that our proposed method can increase the total system satisfaction utility by nearly 20% and reduce the system energy consumption by nearly 10% compared to the benchmark methods.**

***Index Terms*—Computation offloading, mobile edge computing, edge server, resource allocation, game theory.**

I. Introduction

IN THE past decade, the number of mobile devices has experienced an exponential increase, which poses significant challenges to various computing-intensive and time-sensitive applications in wireless networks, such as video conferencing, autonomous driving, and virtual-reality gaming [1], [2]. Such networks often exhibit resource-constrained features of wireless medium and energy storage [3]–[5]. Currently, the integration of cloud computing and edge computing has been widely adopted, and leveraging nearby edge servers to enhance network computing capabilities is regarded as an efficient and reliable approach. However, it cannot provide the services of real-time transmission and computation for huge amounts of mobile data traffic [6]. This is due to the fact that each edge server in resource-constrained wireless networks (RCWNs) has limited computational resources and its computational capacity is shared by all users nearby in the networks. If a large number of computation offloading tasks are performed on the edge server, the edge server becomes overloaded and may cause server failures. To mitigate this negative impact, we introduce a promising device-to-device (D2D)-based offloading [7]. Notably, idle users (e.g., parked vehicles) present in the Internet of Vehicles (IoV) scenario can act as D2D terminals to assist the edge servers in computation offloading tasks and reduce the computational pressure on the edge servers [8], [9]. Thus, for efficient support of the above applications in RCWNs, it is critical to offload users' computational tasks to nearby edge servers and idle terminals via cellular links and D2D links, respectively [10]–[12].

The computation offloading has been extensively studied in RCWNs (see Related Work in Section II for details). Existing research on computation offloading focuses on edge servers [13]–[15], D2D terminals [16]–[18], and joint edge servers and D2D terminals [9], [19]–[21]. For the edge server-based offloading, these works usually consider resource-sharing edge network scenarios with multi-users. To determine optimal offloading strategy, the optimization models are developed with the constraints of user mobility, channel variations, and computational resource allocation in such scenarios. Various methods (e.g., game theory, reinforcement learning) are proposed to solve these optimization problems. As for the D2D-based offloading, these works fully utilize multi-user devices to offload computational tasks, which can reduce the load on edge servers to support computing-intensive applications. Specifically, any device via D2D communication can directly offload tasks to nearby device without forwarding through base stations. D2D communication is a promising technology to

This work was supported in part by the Ministry of Education Innovation Group Joint Fund under Grant 8091B042222; in part by the Major Program of the National Natural Science Foundation of China under Grant 62394321; in part by the National Natural Science Foundation of China under Grant 62372076; in part by the Education Department Research Foundation of Anhui Province under Grant DTR2023051; in part by the Innovation Research Team on Future Network Technology of Chuzhou University; in part by the Innovation Team for Smart Home Appliance Security and Applications of Chuzhou City; in part by ICTFICIAL Oy, Finland; and in part by the European Union's HE Research and Innovation Program HORIZON-JUSNS-2023 through the 6G-Path Project under Grant 101139172. *(Corresponding authors: Wei Su; Bin Yang.)*

Yuan Yuan is with the Zhongguancun Laboratory, Beijing 100081, China, and also with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: yuan.yuan@bjtu.edu.cn).

Bin Yang is with the School of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, Anhui, China (e-mail: yangbinchi@gmail.com).

Wei Su and Hongke Zhang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: wsu@bjtu.edu.cn, hkzhang@bjtu.edu.cn).

Qi Liu is with the Technology Innovation Center, Smart City Research Institute of China Unicom, Beijing 100048, China (e-mail: liuqi49@chinaunicom.cn).

Tarik Taleb is with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, 44801 Bochum, Germany (e-mail: tarik.taleb@rub.de).

significantly improve network performances, such as computational capacity, transmission latency and network coverage, which has been identified as an important component of 5G and beyond [22], [23].

It is noticed that the above work mainly focuses on resource allocation approaches in static regional scenarios. Under such approaches, each user can be allocated a comparable amount of computational resources. However, in computing-intensive and computing-sparse areas, users' demands for computational resources often differ. The computing-intensive regions require more computational resources, while the computing-sparse regions have relatively low computational requirements. Particularly, these two types of computing regions change dynamically based on user demand. For instance, a computing-sparse region may become computing-intensive as demand increases, and vice versa. As a result, two fundamental issues arise. One is how to allocate computational resources to meet the users' demands in the dynamically changed computing regions. Another is how to enhance offloading capacity and user satisfaction once if edge server cannot satisfy the demand of offloading in the computing-intensive region.

To address these issues, this paper explores a joint offloading approach via D2D and edge servers, which can adaptively allocate computational resources to meet the users' demands in the dynamically changed computing regions. However, the joint approach poses two challenging problems. One problem is how to model dynamically changed computational resources in the computing regions according to users' demands. Another one is how to construct the utility function of user satisfaction, and how to solve the complex optimization problem on the utility function maximization. To this end, we first employ user demand and server failure probability to model the dynamic computing regions. Then, we further use prospect-theoretic user satisfaction to construct the utility function and further propose a game-theoretic approach to solve the challenging optimization problem. The main contributions of this paper can be summarized as follows.

1) We first formulate the computation offloading as a multi-user collaborative resource dynamic management optimization problem that aims to maximize user satisfaction utility function, carefully considering critical issues like the non-uniform distribution of computational resources, the user's risk awareness, and the dynamic changes between computing-intensive regions and computing-sparse regions.
2) We construct a resource management scheme for resource allocation of the edge server based on convex optimization. We further propose a dynamic offloading update strategy achieving the maximum of user satisfaction utility function based on game theory and prospect theory, which satisfies the requirements of users' tasks and load balancing at the network edge.
3) Finally, extensive simulation results are presented to validate the effectiveness of our proposed algorithm and to illustrate the impact of system parameter on system utility. We also conduct the comparison study between our proposed algorithm and some benchmark methods.

The remainder of this paper is organized as follows. Section II summarizes the related work. Section III introduces the system model of the RCWN. Section IV constructs the overhead function and the prospect-theoretic utility function, thus formulating the optimization problem of the satisfaction utility. Section V proposes the GTCOS algorithm to solve the optimization problem. Section VI provides extensive simulation results. Finally, Section VII concludes this paper.

II. Related Work

In RCWNs, mobile edge computing (MEC) is an efficient resource management technology to provide the computation offloading service. The existing computation offloading in MEC mainly depends on edge servers and D2D terminals, which can support computing-intensive and time-sensitive applications [9], [13]–[21].

For the edge server-based offloading, these works usually consider resource-sharing edge network scenarios with multi-users. Based on Q-Learning and convex optimization algorithms, the authors in [13] propose a hybrid optimization approach to achieve better network performance and lower computing cost by optimizing computation offloading and resource allocation in the edge network. The approach verifies the feasibility of MEC distributed offloading and solves the optimization problem through decoupling. The work of [14] aims to minimize the maximum task completion latency among all devices by a joint optimization of service caching, task offloading, communication and computation resource allocation, and vehicle placement, while satisfying the energy consumption constraints of all devices. This approach explores the joint optimization problem of resource control in resource-constrained (especially for mobile terminals' energy constraints) edge networks. In [15], the authors investigate the machine learning-based resource allocation mechanism, which can intelligently balance and share computational resources for MEC servers to meet the quality of service (QoS) of applications. At the same time, it indicates that MEC systems need to meet execution latency while minimizing the energy consumption of Internet of Things (IoT) devices.

As for the D2D-based computation offloading, these works fully utilize D2D terminals to offload computational tasks, which can reduce the load of edge servers to support computing-intensive applications. The work of [16] minimizes the overall computation with the constraints of individual energy and computational capacity at both a local user and multiple helpers in a D2D-enabled MEC system. This approach validates the feasibility of D2D computation offloading, where local users can offload their tasks to multiple helpers via D2D communication. In [17], a mobile device and its nearby devices form a mobile ad-hoc edge cloud allowing the agent to offload its tasks to nearby devices, thereby reducing the computational load of the mobile device. The work of [18] is based on the framework of D2D fog computing to realize the sharing of communication resources and computational resources between mobile devices. The approach takes into account the unpredictability of D2D devices while ensuring optimal energy conservation between systems.

Recently, some work has been devoted to investigating joint D2D and edge server-enabled computation offloading schemes. The work of [9] focuses on the partial computation offloading problem in parked vehicle-based mobile edge computing (PVMEC) systems, which extends the computational capacity of MEC servers by utilizing the resources of parked vehicles (PVs), and formulates it as the system utility maximization problem with a careful consideration of offloading decisions, offloading ratios, and resource allocation. The work of [19] conducts the study of task co-offloading problem that offloads computing-intensive industrial tasks to either MEC servers via cellular links or nearby industrial devices via D2D links. A co-offloading framework is further proposed to integrate migration cost and offloading willingness in D2D-assisted MEC networks. The authors in [20] propose a learning-based energy-efficient task offloading method for delay-sensitive and computing-intensive tasks in Vehicular Collaborative Edge Computing (VCEC). The approach explores the joint offloading problem in resource-constrained edge networks and decomposes the optimization problem into resource allocation and task offloading selection. The authors in [21] propose a multi-agent deep reinforcement learning (MADRL)-based approach to solve the task offloading problem in air-ground cooperative vehicular computational networks (AVC2N) by introducing unmanned aerial vehicles (UAVs) and cooperative vehicles (CVs), while satisfying the ultra-reliable and low-latency communication (URLLC) requirements. The approach divides the computation offloading problem into transmission cost optimization and computational resource allocation problems.

We note that the above studies explore individual or joint computation offloading issues, providing a useful perspective for our research. On the one hand, D2D devices can be an effective complement to edge server-based computation offloading, but the unpredictability and optimal energy conservation of D2D devices need to be concerned. On the other hand, efficient computation offloading schemes can be solved one by one by decoupling into sub-problems. Combined with the concept of Distributed Foundational Models, lightweight multi-modal learning models can be deployed in multiple edge servers, and federated learning technology can be used to realize collaborative model updating, thus improving offloading decision-making and state sensing. This improves the intelligence of offloading decision-making and state-sensing [24], [25].

In our previous work [26], we explored the joint design of D2D and edge server for computation offloading. However, this work only considers a simple case that the computational resources are uniformly distributed regardless of the dynamic change of computing-intensive and computing-sparse regions. Motivated by this observation, this paper proposes a dynamic allocation strategy for computational resources, which can flexibly deploy resources for computing-intensive and computing-sparse regions in RCWNs. To improve the computational capacity in computing-intensive region, this paper conducts a joint design for the D2D and edge server-enabled computation offloading, under which the computational tasks of mobile devices can be offloaded to their nearby devices.

III. System Model

A. Network Model

As shown in Fig. 1, we consider an uplink transmission cellular network consisting of one base station (BS), one edge server, N mobile users, and M idle users. In such a cellular network, the computational requirements are dynamically changing, which corresponds to the computing-sparse and computing-intensive cases. For the former, each mobile user can offload its computational tasks to the edge server deployed on the BS via a cellular link. In the latter case, we consider a joint computation offloading of D2D and edge server, with which each mobile user can offload its computational tasks to the edge server and the idle users via D2D communications. We consider that the edge server can serve multiple mobile users and each idle user can similarly serve multiple mobile users. Users are free to choose either edge server or D2D terminals for computation offloading services. It is worth noting that due to the higher risk of edge server failure in computing-intensive regions, users prioritize other computation offloading methods (i.e., D2D-based offloading) in order to mitigate this risk and ensure higher satisfaction utility.

We consider a multi-channel interference-limited wireless communication environment in which all users share the system's bandwidth as the wireless communication channel. Besides, all wireless channels encounter additive white Gaussian noise with variance σ^2.

Note that the "dynamics" refers to the continuous changes in the state of computing regions, while "variation" specifically denotes the switching process between computing-intensive and computing-sparse regions. The red and blue circular arrows illustrated in Fig. 1 represent this dynamic evolution mechanism of computing regions, and the variation process unfolds as follows.

User behavior triggers changes in the computing region, resulting in transitions between computing-intensive and computing-sparse states. Conversely, changes in the computing region also influence user behavior, thereby forming a feedback loop. Specifically, uncertainty in resource availability (e.g., dynamic fluctuations in the failure probability of edge servers) induces user's risk awareness in their offloading decisions [27], [28]. When users reduce the volume of data offloaded, the load on edge servers decreases, leading to a corresponding drop in their failure probability. As users perceive the edge servers to be operating more reliably, they tend to increase the amount of data offloaded in pursuit of higher satisfaction utility. However, this increased offloading pressure gradually raises the failure probability of edge servers once again.

In the RCWN, we utilize computation offloading services to satisfy the demand for time-sensitive and computing-intensive tasks in the computing region. In this network, each edge server is considered as a core network entity with powerful computational capacity to provide services to users. However, its computational capacity is limited and still cannot satisfy the demand of a large number of users in computing-intensive regions. Therefore, it is crucial to provide computation offloading services to idle D2D terminals in computing-intensive

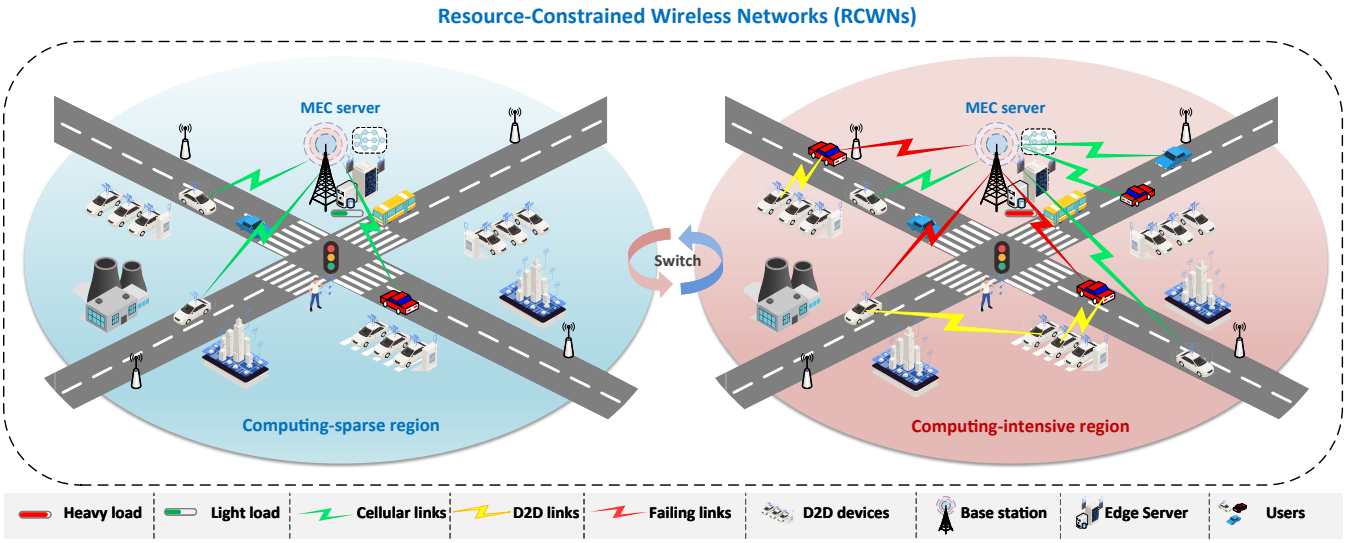


Fig. 1. System scenario

regions. Therefore, users, edge servers, and D2D terminals are jointly involved in the computation offloading process. In addition, we propose a dynamic resource allocation and task offloading strategy to guarantee a higher satisfaction utility of the system. The process is as follows.

When there is a demand for computation offloading services, users need to add small metadata to describe the tasks and update their computational resource usage status. The edge server and all vehicles need to share the required data information and obtain the optimal resource allocation ratio by the Lagrange multiplier method. Then, the edge server provides the corresponding computational resources to the users according to the optimal resource allocation ratio. All users adjust their offloading strategies (i.e., the amount of data offloaded and the way of offloading) by constructing a non-cooperative game. Each user follows the best-response dynamics and loops the above steps (i.e., dynamic resource allocation and task offloading) to select the best response strategy in turn until the system converges to a Nash equilibrium. Based on the resource allocation and task offloading decisions in the Nash equilibrium state, data transmission is completed, and the results are returned to the user devices after being processed by the edge server and the D2D terminal. Note that the RCWN senses the changes in computing-intensive or computing-sparse regions by the server failure probability, and then selects at least one operation from local execution, D2D-based offloading, and server-based offloading.

Deploying the computation offloading solution on the ground is a very challenging endeavor. Computation offloading first requires the deployment of infrastructure with a high-performance edge server on the base station side, and computational resource allocation and management policies are deployed on this server. Then, the base station is responsible for establishing communication links with subscribers and D2D terminals, and at the same time monitoring the load status of the edge server to accomplish the allocation of computational resources and wireless resources.

B. Task Model

We use $\mathcal{N} = \{1, 2, 3, ..., N\}$ to denote the set of N mobile users and use $\mathcal{M} = \{1, 2, 3, ..., M\}$ to denote the set of M idle users available to provide computation offloading services in the computing-intensive region. Note that mobile users and idle users (e.g., parked vehicles) are two different types of vehicles. Among them, idle users are vehicles parked on the roadside or in parking areas based on PVMEC [8], [9]. Mobile users are vehicles that initiate computational requests to fulfill intelligent driving requirements. For the edge server, its available computational capacity is represented as F, and it cannot offload all computational tasks due to limited computational resources. For the edge server ($j = 0$) and any idle terminals ($j \neq 0$), we use d_{ij} to indicate the amount of data offloaded to terminal j, and use d_i to indicate the amount of offloaded data. The offloading ratio of each task on the terminal (i.e., edge server and D2D terminals) is denoted as $\alpha_i = d_i/D_i$, where D_i is the task packet size. Note that $\alpha_i = 1$ represents the complete offloading case and $\alpha_i \in [0, 1)$ represents the partial offloading case. All user tasks are assumed to have identical processing density (in CPU cycles/bit), which is denoted as ρ_i. Thus, the computational task workload is expressed as

$$C_i = D_i \rho_i. \tag{1}$$

C. Load-shifting Model

We use the user's changing demand function R_{tot} to represent the transformation relationship between sparse and intensive computing regions. The computing-intensive and computing-sparse regions correspond to higher and lower user demands, respectively. Based on the user demand function, we introduce the return function R_{re} and the edge server failure probability P_i^f to represent the load-shifting model in

computing-intensive or computing-sparse regions. In particular, the return function illustrates the positive experience of the edge server in providing computational resources, and the edge server failure probability illustrates the likelihood of an edge server being affected by a computing-intensive region and triggering a failure.

The demand function $R_{\rm tot}$ is defined as a continuous and increasing function of the total offloaded data processed by all users [29]. Without loss of generality, any function that follows the above properties can represent the user demand, and an example is used to represent user requirements as

$$R_{\rm tot} = 1/(1 + e^{-\mu \sum_{i \in \mathcal{N}} \rho_i d_{i0}}), \tag{2}$$

where μ is the correction factor for the demand function, and d_{i0} indicates the amount of offloaded data via edge server-based offloading, which reflects the effect of a computing-intensive region or a computing-sparse region on the system utility.

We define the return function $R_{\rm re}$ to be constantly positive, reflecting the positive experience of the edge server in providing computational resources. The return function is a continuous monotonically decreasing concave function [30]. This is due to the fact that the positive experience of return decreases as the total user computational demand increases (i.e., the phenomenon of diminishing marginal returns). Without loss of generality, any function exhibiting the aforementioned properties can represent the positive experience of return, and we choose an example to represent the return function as

$$R_{\rm re}(R_{\rm tot}) = e^{1-R_{\rm tot}} + 1. \tag{3}$$

We define the failure probability P_i^f of an edge server as a function of the value of the user demand. Meanwhile, the failure probability of an edge server should be presented as a convex function in computing-sparse regions with low user demand and a concave function in computing-intensive regions with high user demand [29]. Without loss of generality, we choose a representative example that satisfies convergence to represent failure probability function as

$$P_i^f = R_{\rm tot}^2. \tag{4}$$

The effective probability of the system can be obtained as $1 - P_i^f$, which tends to zero in extremely computing-intensive regions. The model presented in this paper can effectively capture the fundamental trend that servers are more likely to fail as the load increases. At the same time, it simplifies the calculation of mathematical variables (i.e., overhead function, prospect-theoretic utility function and satisfaction utility function) related to server failure probability, ensuring mathematical tractability. This facilitates the resolution of the subsequent optimization problem using game-theoretic approaches.

Note that we illustrate the reasonableness of the model's assumptions about network conditions and user behavior as follows. For user behavior, users tend to choose the computation offloading approach that can obtain higher satisfaction. It should be noted that the demand function of the users is directly related to the amount of offloaded data. The demand function is defined as a continuous and increasing function of the total offloaded data processed by all users [29]. For network conditions, the change of edge server failure probability corresponds to the shift between computing-intensive and computing-sparse regions.

Without loss of generality, we reconcile the single offloading method and the hybrid offloading method by adjusting the server failure probability. To verify the generality of the model, we can flexibly adjust the parameters to reflect the changes in computing regions. the satisfaction utility function validates the user's ability to offload computational tasks in the continuously changing computing region of the RCWN. The return of the system is the positive feedback from the edge servers in providing computational resources to the users, which is closely related to the users' demands and validates the subjectivity of the users, thus avoiding the yield risk in computation offloading. In the optimization problem construction process, we obtain the optimal strategy by maximizing the satisfaction utility. Among them, the satisfaction utility consists of the prospect-theoretic utility provided by edge server-based offloading, local computational consumption, and D2D-based offloading consumption.

D. Performance Model

We define the time and energy consumption required to process computational tasks, which are related to the satisfaction utility function introduced in the next section. The time consumption model should consider local execution and offloading scenarios. Here, the time consumption required for local execution depends on the offloading ratio, the task workload, and the local computational capacity. The time consumption required for offloading depends on link transmission state, task workload, and terminal's computational capacity. The energy consumption model should incorporate local execution energy, data transmission energy, and D2D terminal execution energy. Here, the energy consumption is associated with the computation power parameter, the computational capacity, the task workload, the link transmission state, and the offloading ratio.

We consider that the edge network employs a partial offloading approach to computational tasks. Therefore, the local task execution latency $t_i^{\rm loc}$ is given as

$$t_i^{\rm loc} = (1 - \alpha_i) C_i / f_i^{\rm loc}, \tag{5}$$

where C_i is the workload of the task, α_i is the offloading ratio, and $f_i^{\rm loc}$ is the local computational capacity.

The time consumption on task offloading consists of transmission latency and processing latency, which is expressed as

$$t_{ij}^{\rm off} = t_{ij}^{\rm up} + t_{ij}^{\rm exe}, \tag{6}$$

where $t_{ij}^{\rm up}$ represents transmission latency, and $t_{ij}^{\rm exe}$ represents processing latency.

Transmission latency $t_{ij}^{\rm up}$ of the cellular link or D2D link is given by

$$t_{ij}^{\rm up} = d_{ij} / R_{ij}, \tag{7}$$

where R_{ij} represents the transmission rate.

We consider the cellular and D2D links share the same frequency bands, and thus there exists mutual interference among them. The Rayleigh channel model in small-scale fading is considered in RCWN, i.e., the received signal is assumed to be the sum of a large number of independent and identically distributed multi-path components, which can be approximated as obeying independent Gaussian distributions. Although the impact of time-varying channels on link quality is not explicitly considered in this work, the Rayleigh fading model under fast-fading environments has been extensively used in existing studies to capture such effects [31], [32]. In fact, this paper does account for the intermittent connectivity of communication links. Specifically, we assume that vehicles can access D2D terminals within a certain range. If a vehicle falls outside this range, its computational tasks can only be offloaded to the edge server or executed locally. For a transmitter (e.g., user i) and its receiver (e.g., terminal j), the transmission rate R_{ij} from user i to terminal j is determined as

$$R_{ij} = B \log_2 \left(1 + \frac{p_i h_{ij}^2}{\sigma^2 + \sum_{k \in \mathcal{N}, k \neq i} p_k h_{kj}} \right), \tag{8}$$

where B is the channel bandwidth of the base station, p_i is the signal transmit power of user i for offloading the task to the edge server or idle D2D terminals, h_{ij} denotes the Rayleigh channel coefficient with Gaussian distribution, and σ^2 is the noise power. Here, when $j = 0$, R_{ij} is the transmission rate of the cellular link, and otherwise, it is the transmission rate of D2D link.

The processing latency t_{ij}^{exe} at the terminal (e.g., edge server or idle D2D terminals) is determined as

$$t_{ij}^{\text{exe}} = d_{ij} \rho_i / F_{ij}, \tag{9}$$

where F_{ij} is the computational resource provided by terminal j to user i. The subscript $j = 0$ denotes the computational capacity allocated by edge server, which can be expressed as

$$F_{i0} = \beta_i F, \tag{10}$$

where F represents the computational capacity of the edge server, and β_i represents the ratio of the edge server's computational resources allocated to user i.

Therefore, the offloading latency of the computational task is given by

$$t_{ij}^{\text{off}} = t_{ij}^{\text{up}} + t_{ij}^{\text{exe}} = \frac{d_{ij}}{R_{ij}} + \frac{d_{ij} \rho_i}{F_{ij}}. \tag{11}$$

The energy consumption in RCWN comprises three main components:

1) Local computation energy E_i^{loc}: The energy consumed by mobile user i to process tasks locally.

2) Transmission energy E_{ij}^{tra}: The energy consumed for data transmission from user i to terminal j.

3) D2D computation energy E_{ij}^{com}: The energy consumed by D2D terminal j to execute tasks offloaded from user i.

Note that the edge server is directly connected to the power supply, and thus the energy consumption of the edge server is not considered. The feedback of the results is so small that its transmission time can be ignored. Therefore, the user energy consumption E_i is expressed as

$$E_i = E_i^{\text{loc}} + \sum_{j=0}^{M} E_{ij}^{\text{tra}} + \sum_{j=1}^{M} E_{ij}^{\text{com}}. \tag{12}$$

Three types of energy consumption are denoted by formulas as follows (13)-(15) [33]. Here, the value of E_i^{loc} is the multiplication of the computation power parameter, the cube of the local computational capacity, and the local task execution latency. Therefore, E_i^{loc} is given by

$$E_i^{\text{loc}} = \epsilon \left(f_i^{\text{loc}} \right)^3 t_i^{\text{loc}}, \tag{13}$$

where ϵ is the computation power parameter, f_i^{loc} is the local computational capacity, and t_i^{loc} is the local task execution latency.

The value of E_{ij}^{tra} is the multiplication of the user's signal transmit power and the transmission latency between user i and terminal j. Therefore, E_{ij}^{tra} is given by

$$E_{ij}^{\text{tra}} = p_i t_{ij}^{\text{up}} = p_i d_{ij} / R_{ij}, \tag{14}$$

where p_i is the user's signal transmit power and t_{ij}^{up} is the transmission latency between user i and terminal j.

The value of E_{ij}^{com} is the multiplication of the computation power parameter, the cube of the local computational capacity, and the processing latency of the terminal (e.g., edge server or idle D2D terminals). Therefore, E_{ij}^{com} is given by

$$E_{ij}^{\text{com}} = \epsilon \left(F_{ij} \right)^3 t_{ij}^{\text{exe}}, \tag{15}$$

where ϵ is the computation power parameter, F_{ij} is the local computational capacity, and t_{ij}^{exe} is the processing latency of the terminal (e.g., edge server or idle D2D terminals).

Note that D2D terminals are limited in power supply. Computation offloading may exhaust the energy of the D2D terminals, resulting in the vehicle being unable to perform its duties, such as starting the vehicle engine [9]. Therefore, a constraint is introduced to ensure that the residual battery energy of the D2D terminal (i.e., E_j^{res}) after D2D-based offloading is not less than a specified threshold value $E_{\max}$, which is given by

$$E_j^{\text{res}} = E_j^{\text{cur}} - E_{ij}^{\text{com}} \geq E_{\max}, \ \ \forall i \in \mathcal{N}, j \in \mathcal{M}, \tag{16}$$

where E_j^{res} is the residual battery energy of the D2D terminal, E_j^{cur} is the current energy level of D2D terminal j, and E_{ij}^{com} is the energy consumption of D2D terminal j.

The symbols used in this paper are described in Table I.

IV. Problem Formulation

A. Overhead Function

For any user i, we give three types of overhead functions according to the execution location of its computational tasks [30], [31], [34].

1) ***Case 1 for local overhead:*** Local overhead consists of the time consumption and energy consumption induced by local computing of the unoffloaded data. Then, the local overhead function K_i^{loc} can be expressed as

$$K_i^{\text{loc}} = t_i^{\text{loc}} + E_i^{\text{loc}}. \tag{17}$$

TABLE I
DEFINITION OF MATHEMATICAL SYMBOLS

Symbol	Definition
N	The number of mobile users.
M	The number of idle users.
$\mathcal{N}$	The set of mobile users.
$\mathcal{M}$	The set of idle users.
$\mathcal{S}$	The distribution of offloading strategies.
$\mathbf{d}_{-i}$	The set of offloading strategies (except for user i).
d_i	The offloaded data of user i.
d_{ij}	The offloaded data from user i to D2D terminal j.
D_i	Task packet size.
ρ_i	Processing density.
C_i	The Workload of the task.
α_i	The offloading ratio of each task on the terminal.
β_i	The ratio of the edge server's computational resources allocated to user i.
R_{re}	Return function.
R_{tot}	Demand function.
P_i^f	The failure probability of the edge server.
t_i^{loc}	Local task execution latency.
f_i^{loc}	Local computational capacity.
R_{i0}	Transmission rate of cellular link.
R_{ij}	Transmission rate of D2D link.
h_{ij}	Channel gain.
σ^2	Variance of noise power.
t_{ij}^{up}	Transmission latency between user i and terminal j.
t_{ij}^{exe}	Processing latency of the terminal (e.g., edge server or idle D2D terminals).
t_{ij}^{off}	Offloading latency of the computational task.
F	Computational capacity of the edge server.
F_{ij}	Computational capacity provided by terminal j to user i.
E_i	Energy consumption of user i.
E_i^{loc}	Local computation energy.
E_{ij}^{tra}	Transmission energy.
E_{ij}^{com}	D2D computation energy.
p_i	Signal transmit power of user i.
K_i^{loc}	Overhead function for computing locally.
K_i^{dev}	Overhead function for D2D-based offloading.
K_i^{edg}	Overhead function for edge server-based offloading.
$\mathcal{U}_i$	Prospect-theoretic utility of user i.
$\mathbb{U}_i$	Satisfaction utility function of user i.
μ	Correction factor of demand.
$\kappa, \eta_i, \epsilon_i$	Lagrange multipliers.
λ	Loss aversion parameter.
ω	The weight of potential function.
m, n	Sensitivity parameters.
$T_i^{\max}$	Tolerated latency of user i.
$g_{\max}$	Maximum iteration cycles.

2) ***Case 2 for D2D's overhead:*** D2D's overhead consists of the time consumption and energy consumption induced by offloaded data transmission and computing at the D2D terminal. Then, the overhead function K_{ij}^{dev} for user i offloading to D2D terminal j can be expressed as

$$K_{ij}^{\mathrm{dev}} = t_{ij}^{\mathrm{up}} + t_{ij}^{\mathrm{exe}} + E_{ij}^{\mathrm{tra}} + E_{ij}^{\mathrm{com}}. \tag{18}$$

Therefore, D2D's overhead K_i^{dev} is given by

$$K_i^{\mathrm{dev}} = \sum_{j \in \mathcal{M}} K_{ij}^{\mathrm{dev}}. \tag{19}$$

3) ***Case 3 for edge server's overhead:*** Because of the fragility (i.e., uncertainty of failure) of the edge server as a common pool of resources, the overhead needs to be discussed in a case-by-case manner. Assuming that there is no failure of the edge server, the overhead includes the consumption of the transmission, the time consumption of the server computing, and the system return. Then, the overhead function K_i^{edg} can be expressed as

$$K_i^{\mathrm{edg}} = t_{i0}^{\mathrm{tra}} + \frac{d_{i0}\rho_i}{F_{i0}} + E_{i0}^{\mathrm{tra}} - d_{i0}R_{re}. \tag{20}$$

Assuming that the edge server fails, users will compute locally, but the consumption caused by the transmission has already been incurred. Then, the overhead function K_i^{edg} can be expressed as

$$K_i^{\mathrm{edg}} = t_{i0}^{\mathrm{tra}} + E_{i0}^{\mathrm{tra}} + K_i^{\mathrm{loc}}. \tag{21}$$

We assume that the probability of failure is P_i^f (see Section III.C). Then, the mathematical expectation of edge server's overhead can be expressed as

$$\begin{aligned}\mathbb{E}(K_i^{\mathrm{edg}}) =& (1 - P_i^f)\left(t_{i0}^{\mathrm{tra}} + \frac{d_{i0}\rho_i}{F_{i0}} + E_{i0}^{\mathrm{tra}} - d_{i0}R_{re}\right) \\ &+ P_i^f\left(t_{i0}^{\mathrm{tra}} + E_{i0}^{\mathrm{tra}} + K_i^{\mathrm{loc}}\right).\end{aligned} \tag{22}$$

B. Prospect-theoretic Utility Function

In order to address the uncertainty of edge server failure and the risk awareness of users in offloading decisions, and considering that real-life users are not risk-neutral, we adopt the principles of Prospect Theory [35]. Prospect theory was proposed by *D. Kahneman* and *A. Tversky*, which is a behavioral model in which users make decisions under the risk and uncertainty of the benefits associated with their choices [30], [34]. The user's losses and gains in prospect theory are evaluated based on reference points. The user's relevant utility function is concave for gains (i.e., the user is risk averse in gains) and convex for losses (i.e., the user seeks risk in losses).

Based on overhead functions and prospect theory, we construct the prospect-theoretic utility function as

$$\mathcal{U}_i = \begin{cases} \left(K_i^{\mathrm{ref}} - K_i^{\mathrm{edg}}\right)^m, & \text{w.p. } 1 - P_i^f \\ -\lambda\left(K_i^{\mathrm{edg}} - K_i^{\mathrm{ref}}\right)^n, & \text{w.p. } P_i^f \end{cases}, \tag{23}$$

where the reference value of user overhead $K_i^{\mathrm{ref}} = K_i^{\mathrm{loc}}$, P_i^f is the failure probability of edge server, m represents the sensitivity parameter of the yield growth function, n represents the sensitivity parameter of the loss growth function, and λ represents the loss aversion parameter. And, we denote "with probability" as "w.p." for brevity. Then, the mathematical expectation of the prospect-theoretic utility can be expressed as

$$\begin{aligned}\mathbb{E}(\mathcal{U}_i) =& (1 - P_i^f)\left(K_i^{\mathrm{loc}} - t_{i0}^{\mathrm{tra}} - \frac{d_{i0}\rho_i}{F_{i0}} - E_{i0}^{\mathrm{tra}} + d_{i0}R_{re}\right)^m \\ &- \lambda P_i^f\left(t_{i0}^{\mathrm{tra}} + E_{i0}^{\mathrm{tra}}\right)^n.\end{aligned} \tag{24}$$

C. Optimization Problem of Satisfaction Utility

The satisfaction utility demonstrates the user satisfaction with computation offloading, which consists of the prospect-theoretic utility provided by edge server-based offloading, local computational consumption, and D2D-based offloading

consumption. Thus, the satisfaction utility $\mathbb{U}_i$ can be expressed as

$$\mathbb{U}_i = \mathbb{E}\left(\mathcal{U}_i\right) - K_i^{\mathrm{loc}} - K_i^{\mathrm{dev}}, \tag{25}$$

where $\mathbb{E}\left(\mathcal{U}_i\right)$ represents the mathematical expectation of the edge server's overhead in *Case 3*, K_i^{loc} represents the local overhead in *Case 1*, and K_i^{dev} represents the D2D's overhead in *Case 2*. Combining the prospect-theoretic utility function proposed in [34] with network model, task model, load-shifting model, and performance model, the satisfaction utility of the RCWN can be given by

$$\begin{aligned}\mathbb{U}_i =&(1-P_i^f)\left(K_i^{\mathrm{loc}} - t_{i0}^{\mathrm{tra}} - \frac{d_{i0}\rho_i}{F_{i0}} - E_{i0}^{\mathrm{tra}} + d_{i0}R_{re}\right)^m \\ &- \lambda P_i^f\left(t_{i0}^{\mathrm{tra}} + E_{i0}^{\mathrm{tra}}\right)^n - \left(t_i^{\mathrm{loc}} + E_i^{\mathrm{loc}}\right) \\ &- \sum_{j\in\mathcal{M}}\left(t_{ij}^{\mathrm{up}} + \frac{d_{ij}\rho_i}{F_{ij}} + E_{ij}^{\mathrm{tra}} + E_{ij}^{\mathrm{com}}\right).\end{aligned} \tag{26}$$

Our objective is to maximize the satisfaction utility function of each user, which can be formulated as

$$\begin{aligned}
\textbf{\textit{P1:}}\quad & \max_{\boldsymbol{d_i},\boldsymbol{\beta_i}} \ (\mathbb{U}_i) && (27)\\
\text{s.t.}\quad & \mathbb{E}(\mathcal{U}_i) \geq 0, \ \forall i \in \mathcal{N}, && (27a)\\
& t_{ij}^{\mathrm{off}} \leq T_i^{\max}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}\cup\{0\}, && (27b)\\
& \beta_i \in [0,1], \ \forall i \in \mathcal{N}, && (27c)\\
& \sum_{i\in\mathcal{N}} \beta_i \leq 1, && (27d)\\
& E_j^{\mathrm{res}} \geq E_{\max}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}. && (27e)
\end{aligned}$$

Here, $\boldsymbol{d_i}$ represents the distribution of offloading strategies (i.e., the amount of offloaded data), $\boldsymbol{\beta}_i$ represents the set of resource allocation ratios, $T_i^{\max}$ represents the tolerated latency of user i, and $j \in \{0\}$ represents the approach of edge server-based offloading. The constraint in (27a) indicates that the prospect-theoretic utility is always positive. (27b) indicates the tolerance level of system latency. (27c) and (27d) ensure that the total computational resource allocation of one user or all users does not exceed the edge server capacity. Finally, (27e) ensures that the residual battery energy E_j^{res} of the D2D terminal after D2D-based offloading is not less than a specified threshold value $E_{\max}$.

Note that two types of utility functions are used in this paper, i.e., prospect-theoretic utility function and satisfaction utility function. The prospect-theoretic utility function characterizes users' valuation of available offloading options during decision-making, incorporating both the inherent uncertainty and subjective perception of potential gains when choosing between local computing and offloading. This function comprises two key elements, i.e., the gain component and the loss component, both derived from prospect theory principles. The satisfaction utility function quantifies users satisfaction with the system computation offloading performance, and consists of the prospect-theoretic utility provided by the local overhead, the D2D's overhead, and the edge server's overhead.

It should be noted that network density in this paper refers to the size of the amount of computational data in a single region, and the change in network density is reflected as a transition between computing-intensive and computing-sparse regions. User demand refers to the computation offloading service required by a single user to accomplish time-sensitive and computing-intensive tasks in a computing region. Higher user demand implies that the amount of data to be offloaded is larger, and the corresponding computing region will shift to a computing-intensive region. Computing-intensive regions require a large number of computational resources and high-performance devices, and limited resources often bring more uncertainty. At this time, the failure probability of the edge server increases. Continuing to offload data to the edge server will significantly reduce the user satisfaction utility. To guarantee the optimal satisfaction utility, we dynamically adjust the resource allocation rate and the offloading policy. Each user follows the best-response dynamics, completes the data transmission based on the optimal decision in the Nash equilibrium, and finally returns the result to the user device.

V. Utility Function Maximization

A. Problem Decoupling

Since the optimization problem in Eq. (27) includes integer-constrained variables, continuous variables, and nonlinear terms, the problem is a mixed integer nonlinear programming (MINLP) problem. This is a NP-hard problem, which is difficult to solve directly using traditional optimization methods. To solve the MINLP problem, we decompose this complex problem into two sub-optimization problems. The first one is the optimization problem of computational resource allocation, where we use a Lagrange multiplier method to optimize the resource allocation scheme on the computational resource allocation ratio. The second one is the updating problem of the computation offloading strategy, where we design a game-theoretic algorithm for strategy updating.

We first optimize the computational resource allocation policy to reduce the computational time consumption of the edge server. Notably, we find that reducing the computational time consumption on the edge server (i.e., index $j = 0$) can increase the satisfaction utility based on Eq. (26). Thus, the first sub-problem ***P2*** can be formulated as

$$\begin{aligned}
\textbf{\textit{P2:}}\quad & \min_{\boldsymbol{\beta_i}} \ t_{i0}^{\mathrm{exe}} = \min_{\boldsymbol{\beta_i}} \frac{d_{i0}\rho_i}{\beta_i F} && (28)\\
\text{s.t.}\quad & \beta_i \in [0,1], \ \forall i \in \mathcal{N}, && (28a)\\
& \sum_{i\in\mathcal{N}} \beta_i \leq 1, && (28b)
\end{aligned}$$

where (28a) ensures that the total allocation of computational resources does not exceed the capacity of the edge server, and (28b) indicates the constraint on the computational resource allocation ratio.

The second sub-optimization problem ***P3*** on the computation offloading strategy can be formulated as

$$\begin{aligned}
\textbf{\textit{P3:}}\quad & \max_{\boldsymbol{d_i}} \ (\mathbb{U}_i) && (29)\\
\text{s.t.}\quad & \mathbb{E}(\mathcal{U}_i) \geq 1, \ \forall i \in \mathcal{N}, && (29a)\\
& t_{ij}^{\mathrm{off}} \leq T_i^{\max}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}\cup\{0\}, && (29b)\\
& E_j^{\mathrm{res}} \geq E_{\max}, \ \forall i \in \mathcal{N}, j \in \mathcal{M}, && (29c)
\end{aligned}$$

where (29a) indicates that the prospect-theoretic utility is always positive, (29b) indicates the tolerance level of system latency, and (29c) ensure that the residual battery energy E_j^{res} of the D2D terminal after D2D-based offloading is not less than the specified threshold.

B. Resource Allocation Based on Convex Optimization

To address the optimization problem ***P2***, we construct a resource allocation method based on convex optimization to obtain the optimal allocation of computational resources for each user. We transform the optimization problem ***P2*** into $R(\beta_1, \beta_2, \ldots, \beta_N)$ and three constraints as

$$R(\beta_1, \beta_2, \ldots, \beta_N) = \min_{\boldsymbol{\beta_i}} \sum_{i=1}^{N} t_{i0}^{\text{exe}} = \min_{\boldsymbol{\beta_i}} \sum_{i=1}^{N} \frac{d_{i0}\rho_i}{\beta_i F} \tag{30}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \beta_i - 1 \leq 0, \tag{30a}$$

$$-\beta_i \leq 0, i \in \mathcal{N}, \tag{30b}$$

$$\beta_i - 1 \leq 0, i \in \mathcal{N}, \tag{30c}$$

where (30a) ensures that the total computational resource allocation of all users does not exceed the edge server capacity, (30b) and (30c) denote the upper and lower bounds of the ratio of computational resources that can be allocated to a single user, respectively.

We construct the Hessian matrix representation of problem ***P2*** by taking the second order derivative as

$$\mathcal{H} = \begin{bmatrix} 2d_{10}\rho_i/\beta_1^3 F & 0 & \cdots & 0 \\ 0 & 2d_{20}\rho_i/\beta_2^3 F & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2d_{N0}\rho_i/\beta_N^3 F \end{bmatrix}, \tag{31}$$

where each element of the above matrix is a non-negative real number. Therefore, $\mathcal{H}$ is a positive definite matrix. Since the positive definite matrix and the constraints are linear in Eq. (30), it is a convex optimization problem. According to the theory of convex optimization, the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient, and finding a solution that satisfies the KKT conditions is the global optimal solution [36]. We use the Lagrange multiplier method with KKT conditions to solve the optimization problem. The Lagrange function can be expressed as

$$\begin{aligned} \mathcal{L}(\beta_i, \delta) = &\sum_{i=1}^{N} \frac{d_{i0}\rho_i}{\beta_i F} + \kappa \left(\sum_{i=1}^{N} \beta_i - 1 \right) \\ &+ \sum_{i=1}^{N} \eta_i (-\beta_i) + \sum_{i=1}^{N} \xi_i (\beta_i - 1), \end{aligned} \tag{32}$$

where κ, η_i, and ξ_i are the Lagrange multipliers. The KKT conditions are given by

$$\begin{cases} -d_{i0}\rho_i/\beta_i^2 F + \kappa - \eta + \xi = 0, \\ \kappa \left(\sum_{i=1}^{N} \beta_i - 1 \right) = 0, \ \ \sum_{i=1}^{N} \beta_i - 1 \leq 0, \\ \sum_{i=1}^{N} \eta (-\beta_i) = 0, \ \ -\beta_i \leq 0, \\ \sum_{i=1}^{N} \xi (\beta_i - 1) = 0, \ \ \beta_i - 1 \leq 0, \\ \kappa \geq 0, \ \ \eta \geq 0, \ \ \xi \geq 0. \end{cases} \tag{33}$$

By combining the stationarity condition, the primal feasibility condition, the dual feasibility condition, and the complementary slackness condition from Eq. (33), and assuming that the equality constraint is active (i.e., $\sum_{i=1}^{N} \beta_i = 1$), the internal solution can be simplified as follows.

$$\kappa = \frac{\left(\sum_{i=1}^{N} \sqrt{d_{i0}\rho_i} \right)^2}{F}, \ \ i \in \mathcal{N}. \tag{34}$$

Further, we obtain the closed-form solution (i.e., optimal resource allocation ratio) under KKT conditions as

$$\beta_i^* = \sqrt{\frac{d_{i0}\rho_i}{\kappa F}} = \frac{\sqrt{d_{i0}\rho_i}}{\sum_{i=1}^{N} \sqrt{d_{i0}\rho_i}}, i \in \mathcal{N}. \tag{35}$$

Based on the closed-form expression in Eq. (35), we can see that the optimal resource allocation ratio β_i^* is closely related to the amount of user's offloaded data. As the amount of user's offloaded data increases, the optimal resource allocation ratio β_i^* also increases.

C. Computation Offloading Strategy

We propose a computation offloading strategy based on a non-cooperative game theory. We use $\mathbb{G} = \{\mathcal{N}, \mathcal{S}, \mathcal{Z}\}$ to represent the game model. Here, $\mathcal{N}$ represents the set of mobile users participating in the game, the total strategy space $\mathcal{S}$ represents the strategy distribution of users in a Cartesian coordinate system, i.e. $\mathcal{S} = d_1 \times d_2 \times \ldots \times d_{N-1} \times d_N$, where $d_i = \{d_{i0}, d_{i1}, \ldots, d_{ij}\}, i \in \mathcal{N}$, and $\mathcal{Z}$ represents the total system satisfaction utility, which is the sum of the satisfaction utility values of all participants, i.e. $\mathcal{Z} = \sum_{i \in \mathcal{N}} \mathbb{U}_i$.

The game aims to obtain an appropriate strategy to achieve the Nash equilibrium condition, under which the decision change of each participant will not affect the satisfaction utility function [37]. For a given state, the potential function of the game represents the trend of utility change among all users with different strategies offloading to the edge server or an idle D2D terminal. For a potential game, we have

$$\mathcal{Z}\Big(d_i, \mathbf{d}_{-i}\Big) - \mathcal{Z}\Big(d_i', \mathbf{d}_{-i}\Big) = \omega\Big(\Phi\Big(d_i, \mathbf{d}_{-i}\Big) - \Phi\Big(d_i', \mathbf{d}_{-i}\Big)\Big), \tag{36}$$

where d_i represents the offloading strategy of user i, d_i' represents the updated strategy of user i, $\mathbf{d}_{-i}$ represents the set of offloading strategies except for user i, ω represents the weight of potential function, $\Phi(d_i, \mathbf{d}_{-i})$ represents the potential function, and $\mathcal{Z}(d_i, \mathbf{d}_{-i})$ represents the total system satisfaction utility (i.e., the sum of satisfaction utility values for all users) with strategies d_i and $\mathbf{d}_{-i}$.

Consider an exact potential game, i.e., $\omega = 1$. Then,

$$\mathcal{Z}\Big(d_i, \mathbf{d}_{-i}\Big) - \mathcal{Z}\left(d_i', \mathbf{d}_{-i}\right) = \Phi\Big(d_i, \mathbf{d}_{-i}\Big) - \Phi\left(d_i', \mathbf{d}_{-i}\right). \tag{37}$$

The above equation shows that when the user's strategy is updated, the changes in the total system satisfaction utility

function and the potential function are equal. The difference of the potential function is expressed as

$$\begin{aligned}
&\Phi\left(d_i', \mathbf{d}_{-i}\right) - \Phi\left(d_i, \mathbf{d}_{-i}\right) \\
=&\sum_{i\in\mathcal{N}, d_i\in\left(d_i', \mathbf{d}_{-i}\right)} \mathbb{U}_i\left(d_i\right) - \sum_{i\in\mathcal{N}, d_i\in\left(d_i, \mathbf{d}_{-i}\right)} \mathbb{U}_i\left(d_i\right) \\
=&\mathbb{U}_i\left(d_i'\right) - \mathbb{U}_i\left(d_i\right) + \sum_{i\in\mathcal{N}, d_i\in\mathbf{d}_{-i}} \mathbb{U}_i\left(d_i\right) - \sum_{i\in\mathcal{N}, d_i\in\mathbf{d}_{-i}} \mathbb{U}_i\left(d_i\right) \\
=&\mathbb{U}_i\left(d_i'\right) - \mathbb{U}_i\left(d_i\right),
\end{aligned} \tag{38}$$

where $\mathbb{U}_i\left(d_i\right)$ represents the satisfaction utility of user i with the offloading strategy d_i. Therefore, the proposed game is an exact potential game (EPG) and satisfies the conditions of a Nash equilibrium.

We then propose a computation offloading strategy named game-theoretic cooperative offloading strategy (GTCOS), which is introduced in **Algorithm 1**. The GTCOS algorithm is summarized as follows.

According to the state of RCWNs, we first calculate the optimal computational resources to be allocated at each step, i.e.

$$F_{i0}^* = \beta_i^* F, \tag{39}$$

where β_i^* represents the optimal resource allocation ratio.

Then, the user iterates the strategy (e.g., edge server-based strategy or D2D-based strategy) and observes the change of the utility function by game theory. The user takes its maximum value as the best response and uses the strategy at this point as the iterative strategy. This procedure is repeated until the strategy satisfies the Nash equilibrium condition (i.e., every user has no more updated responses). The iterative strategy d_i is given by

$$d_i = \begin{cases} d_i', & \text{if } \mathbb{U}_i(d_i', \mathbf{d}_{-i}) > \mathbb{U}_i(d_i, \mathbf{d}_{-i}), i \in \mathcal{N}, \\ d_i, & \text{otherwise.} \end{cases} \tag{40}$$

The set of optimal offloading strategy d_i^* is given by

$$d_i^* = d_i = \{d_{i0}, d_{i1}, \ldots, d_{ij}\}, i \in \mathcal{N}, \tag{41}$$

and Nash equilibrium offloading strategies $\mathcal{S}^*$ are given by

$$\mathcal{S}^* = d_1^* \times d_2^* \times \ldots \times d_{N-1}^* \times d_N^*. \tag{42}$$

It should be noted that although computation offloading exhibits strong potential for practical deployment in RCWNs, its large-scale adoption remains dependent on the continuous advancement of communication and computing infrastructures, as well as breakthroughs in key networking technologies. Therefore, the evolution of computation offloading should be regarded as an incremental process that co-develops with the maturity of the underlying infrastructure.

In practical vehicular networking scenarios, network topologies are inherently complex and highly dynamic, which introduces several challenges to real-world implementation. Communication links between vehicles are often affected by occlusions caused by obstacles such as buildings, bridges, and large vehicles, particularly in dense urban areas, leading to intermittent connectivity and D2D signal degradation. Moreover, frequent handovers resulting from high vehicular mobility can exacerbate communication instability, causing link interruptions, transmission latency, and potential packet loss. Additionally, the operation of RCWNs requires the collection of real-time vehicular data, such as location and driving behavior, which raises serious concerns about user privacy and data security. Ensuring secure, anonymous, and privacy-preserving data exchange in such a dynamic and decentralized environment remains a significant challenge for practical implementation.

Algorithm 1: GTCOS: A game-theoretic cooperative offloading strategy.

Input: mobile users set $\mathcal{N} = \{1, 2, 3, ..., N\}$, idle users set $\mathcal{M} = \{1, 2, 3, ..., M\}$, offloading strategies $\mathcal{S} = d_1 \times d_2 \times \ldots \times d_{N-1} \times d_N$, game model $\mathbb{G} = \{\mathcal{N}, \mathcal{S}, \mathcal{Z}\}$, iteration cycles $g = 0$, maximum iteration cycles $g_{\max}$, and other algorithm parameters.

Output: optimal allocation of computational resource F_{i0}^* and Nash equilibrium offloading strategies $\mathcal{S}^*$.

```
initialization;
while each user i, i ∈ N do
    calculate β_i^* and F_{i0}^* by Lagrange multiplier
      method (see Section V.B);
    obtain initialized offloading strategy S and
      F_ij, j ∈ M ∪ {0};
    calculate U_i(d_i, d_{-i}) by Eq.(26) and Eq.(35);
    select new strategy d_i' satisfies
      U_i(d_i', d_{-i}) = argmax_{j∈M∪{0}} U_i(d_i, d_{-i});
    if U_i(d_i', d_{-i}) > U_i(d_i, d_{-i}),   ∀i ∈ N then
        update d_i = d_i';
    else
        d_i = d_i;
    end
    g = g + 1;
end
repeat steps 2-13 until the algorithm converges or the
  number of iteration cycles g = g_max;
return d_i^* = d_i;
obtain Nash equilibrium offloading strategies
  S^* = d_1^* × d_2^* × ... × d_{N-1}^* × d_N^*.
```

D. Complexity Analysis

We generate the user's computation offloading policy via a distributed iterative low-complexity algorithm. For the optimization problem ***P2***, the computational complexity depends on the complexity of the Lagrange multiplier method and the number of algorithm iteration cycles. According to Eq. (35), the solution process involves traversing N elements for square root and summation operations, so the computational complexity of the solutions for optimal resource allocation is $O(N)$, where N is the number of users involved in resource allocation. Thus, the computational complexity is $O(N \times g)$. For the optimization problem ***P3***, we perform the following analysis. According to the best-response dynamics, we denote

TABLE II
THE SIMULATION PARAMETERS

Parameters	Value
Transmission channel bandwidth	10 MHz
computational capacity of user	$[5, 7] \times 10^9$ CPU cycles/s
computational capacity of edge server	3×10^{10} CPU cycles/s
Signal transmit power of user	$\{10, 30\}$ dBm
Gaussian white noise power	-114 dBm
Packet data size	$[10, 20]$ Mb
Processing density	100 CPU cycles/bit
Tolerated latency of user	$[0.5, 1]$s
Sensitivity parameter	$\{0.1, 0.2, \cdots, 0.9, 1.0\}$
Loss aversion parameter	$\{0.2, 0.4, \cdots, 1.8, 2.0\}$
Correction factor of demand function	1.5×10^{-10}

the computational complexity of a single convex optimization solution as $O(A)$. Considering that there are N users in the RCWN involved in the computation offloading, and at the same time the algorithm needs g iteration cycles to converge to the Nash equilibrium, the computational complexity is $O(N \times g \times A)$. In summary, the computational complexity of the GTCOS algorithm we proposed is $O(N \times g \times (A+1))$. Since $O(A)$ is much larger than $O(1)$, the computational complexity is reduced to $O(N \times g \times A)$.

In large-scale scenarios, to enhance system manageability and computational efficiency, we partition the overall network into several sub-regions and deploy dedicated edge servers within each region, thereby constructing multiple non-overlapping small-scale network sub-scenarios. Within these localized areas, we assume that signaling delay between nodes can be neglected to simplify model analysis and optimization.

VI. SIMULATION AND ANALYSIS

We conduct simulation study to analyze the convergence performance of GTCOS algorithm, and also to illustrate the impact of critical system parameters on the offloaded data and failure probability. We further provide performance comparison between GTCOS algorithm and other methods.

A. Parameter Setting

In this simulation, we consider an RCWN scenario consisting of an edge server at the base station, and 20 randomly distributed mobile users with computational requirements with high-reliable and low-latency tasks (e.g., video conferencing, virtual-reality games, etc.), and 10 idle users are D2D terminals for task offloading. The transmit power of mobile users is set to a constant value selected from the set $\{10, 30\}$. Although the randomness and time variability of task arrivals were not explicitly incorporated during the model construction phase, we fully considered the stochastic nature of task arrivals in the experimental design and simulation process (i.e., the data packet sizes were randomly sampled within a given range), in order to more accurately reflect the behavior of system under dynamic conditions. The detailed parameter setting is listed in Table II.

B. Performance Analysis

*1) **Convergence Performance:*** To ensure the reliability of our proposed GTCOS algorithm, we first analyze its convergence performance. Regarding the case of six mobile users, Fig. 2 shows how the number of iteration cycles affects the amount of offloaded data for each user. It is observed from Fig. 2 that the number of offloaded data first increases, then experiences dynamic changes, and finally keeps unchanged. This is because when the number of iteration cycles is relatively small, each user can obtain gains through offloading its tasks to the edge server, and thus the amount of offloaded data increases as the number of iteration cycles increases. As the number of iteration cycles continues to increase, the user demand and the probability of server congestion increase, which leads to the dynamic changes of the number of offloaded data. For each user, as the number of iteration cycles increases up to a threshold, the state of each user converges to a Nash equilibrium, and thus the amount of offloaded data remains unchanged.

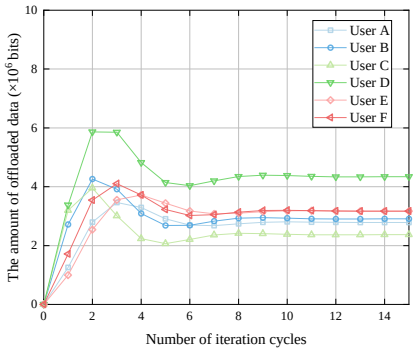


Fig. 2. Variation of offload data for different users.

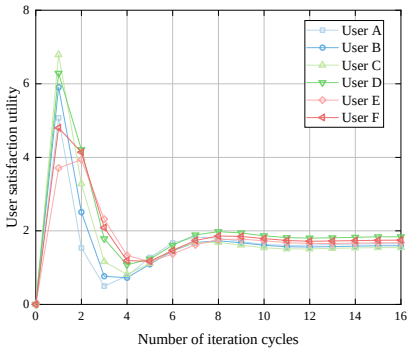


Fig. 3. Variation of satisfaction utility for different users.

Fig. 3 shows how the number of iteration cycles affects the user satisfaction utility. We can see from Fig. 3 that the tread of change is similar to that in Fig. 2. This can be explained as follows. At the beginning of iteration, increasing the number of iteration cycles can increase the satisfaction utility of each user via offloading its tasks to edge server. When the number of iteration cycles becomes bigger, the edge server cannot execute all tasks, and thus these tasks will be executed locally or offloaded to nearby D2D terminal for handling, which leads to the dynamic changes of the number of satisfaction

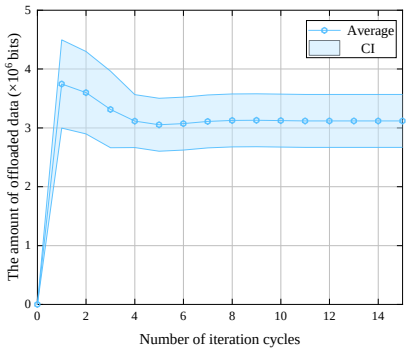


Fig. 4. Average offloaded data convergence for all users.

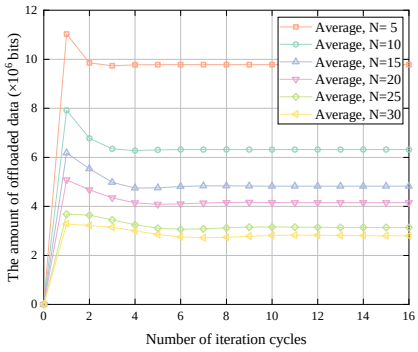


Fig. 6. Variation of average offloaded data with different users.

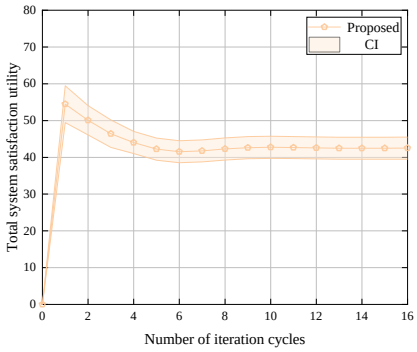


Fig. 5. Total system satisfaction utility convergence of the GTCOS algorithm.

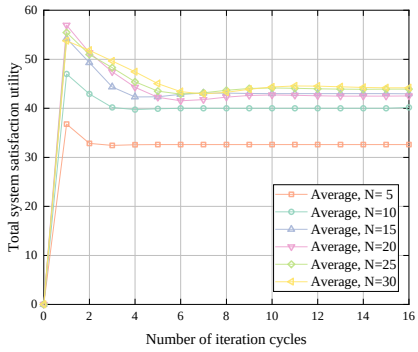


Fig. 7. Variation of total system satisfaction utility with different users.

utility. When the number of iteration cycles further increases, the system converges to the Nash equilibrium and thus the satisfaction utility of each user remains unchanged.

Regarding the case of the total 20 mobile users requiring task offloading, Figs. 4 and 5 show that both the amount of average offloaded data and total system satisfaction utility tend towards constants. This also indicates the convergence of the GTCOS algorithm. Note that the confidence intervals are shaded in these figures and the confidence level is 95%.

Figs. 6 and 7 illustrate the impacts of different numbers of mobile users on the amount of offloaded data and the total system satisfaction utility. Fig. 6 shows that as the number of mobile users increases, individual user can offload less amount of data to the edge server or D2D terminals. This is due to the fact that when too many users enter the computing region, the large number of computational tasks can cause the congestion of resource constrained system. Fig. 7 shows that as the number of users increases, the total system satisfaction utility increases. However, a careful observation from Fig. 7 shows that the average satisfaction utility of individual user decreases as the number of users increases. This is because for limited system computational resources, the allocated resources of individual user decrease with the increase of the number of users.

The curves in Figs.2 to 7 illustrate that the system satisfaction utility (i.e., the sum of user satisfaction utility) varies with system parameters. Note that the user satisfaction utility is modeled based on prospect theory from economics. Within the framework of prospect theory, users are risk-aware. Specifically, due to concerns about potential failures of edge servers, users tend to proactively avoid risks when making decisions.

In the initial phase, the probability of edge server failure in computing-sparse regions is relatively low, and users, driven by risk perception, tend to prefer offloading to edge servers. Thus, both the local and D2D's overheads decrease. We can see that the user satisfaction utility ascends as the values of local and D2D's overheads decrease in the second and third items of the user satisfaction utility in Eq. (25). When the computing region becomes related intensive, the edge server has a high failure probability, and thus the users are more inclined to execute tasks locally or via D2D terminals. This phenomenon will lead to an increase in the overhead function, thereby resulting in a decline in the user satisfaction utility.

In summary, the user satisfaction utility exhibits a "rise-then-fall" trend as the system evolves dynamically, eventually converging to a Nash equilibrium. In fact, the final utility has increased from the initial period. This is because, in the initial period, the prospect-theoretic utility made a significant contribution, but due to the substantial increase in the probability of failure, the prospect-theoretic utility declined rapidly. Therefore, from the overall trend of changes in the utility

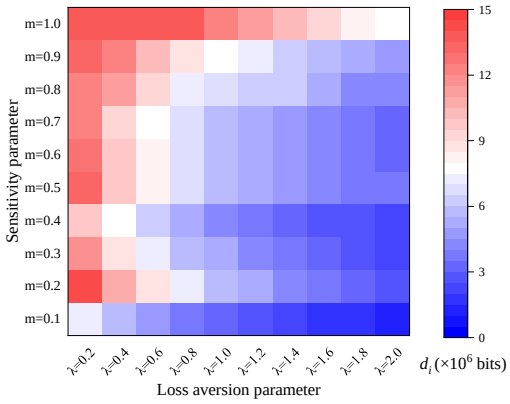


Fig. 8. Offloaded data with different sensitivity parameters and different loss aversion parameters.

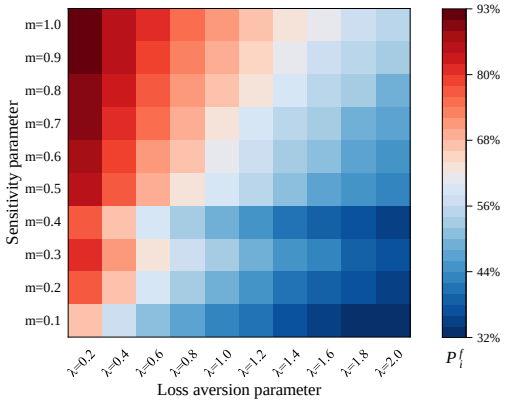


Fig. 9. Failure probability with different sensitivity parameters and different loss aversion parameters.

function, the final utility has increased from the initial period. This process reflects the evolutionary dynamics of the trade-off between risk awareness and resource utilization within the system.

The heat maps in Figs. 8 and 9 show the impacts of the sensitivity parameter m and the loss aversion parameter λ on offloaded data and failure probability. Increasing the sensitivity parameter could increase the yield section of utility in Eq. (23) (i.e., $K_i^{\text{ref}} - K_i^{\text{edg}}$), which motivates users to offload computational tasks to the edge server. However, as the sensitivity parameter increases, the failure probability of the server also increases. When the loss aversion parameter increases, users will reduce the loss to ensure a smaller value of the loss section in Eq. (23) (i.e., $K_i^{\text{edg}} - K_i^{\text{ref}}$), and thus they will more willing to execute the tasks locally rather than offloading them to the edge server. This will decrease the probability of server failure. Note that our method is based on offloading as much data as possible to a high-performance server, while ensuring an acceptable server failure probability. As a result, there is a trade-off on the failure probability by a careful selection for the sensitivity parameter m and the loss

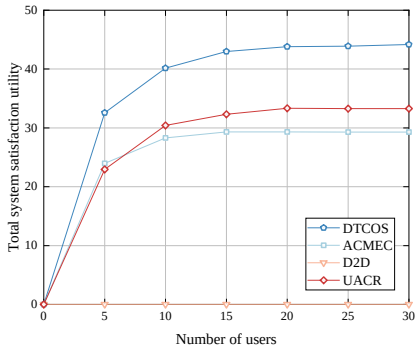


Fig. 10. Impact of the number of users with task requests on total system satisfaction utility.

aversion parameter λ [30], [31]. With careful observation from Figs. 8 and 9, we find that when m=0.4 or m=0.1, the failure probability P_i^f maintains at a low level. Under the settings of the parameters m=0.4 and λ=1.0, we can guarantee a higher value of the satisfaction utility function and also increase the amount of offloaded data with a lower probability of server failure.

*2) **Efficiency Analysis:***

In order to validate the efficiency of our proposed GTCOS algorithm, this section compares and analyzes its performance with the following three benchmark methods.

All centralized in MEC servers strategy (ACMEC): The ACMEC scheme considers that all computation offloading tasks use the edge server-based offloading strategy, and sets the task offloading ratio and resource allocation ratio to fixed values (i.e., $d_i = 0.5 \times D_i$ and $\beta_i = 1/N$).

D2D-based offloading strategy (D2D): The D2D scheme considers that all computation offloading tasks use the D2D terminal-based offloading strategy and sets the task offloading ratio to a fixed value (i.e., $d_i = 0.5 \times D_i$).

Uniform allocation of computational resources strategy (UACR): The UACR scheme considers that all computation offloading tasks use the offloading strategy of the joint edge servers and D2D terminals, and sets the task offloading ratio and the resource allocation ratio to fixed values (i.e., $d_i = 0.5 \times D_i$ and $\beta_i = 1/N$).

Since users have the feature of random mobility, the number of users changes dynamically in the communication range of the edge server. Thus, we explore how the number of users affects the total system satisfaction utility under the fixed setting of the number of idle devices, as shown in Fig. 10. We can see from Fig. 10 that as the number of users increases, the total system utilities increase under these three algorithms (except for the D2D scheme). This is because the positive value of our satisfaction depends in part on the expectations of the edge servers, so there are no expectations for D2D scheme. It is notable that our GTCOS algorithm can increase the total system satisfaction utility nearly by 20% compared to the other three algorithms. It can be explained as follows. As the number of users increases, each user can obtain the edge network resources to improve satisfaction utility and

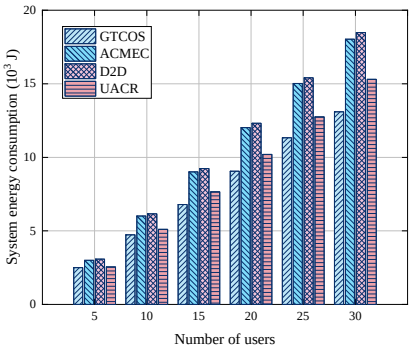


Fig. 11. System energy consumption in different scenarios.

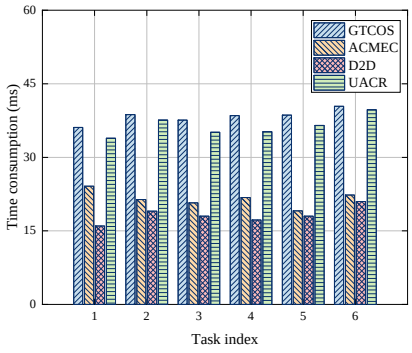


Fig. 12. Time consumption of the single task in different scenarios.

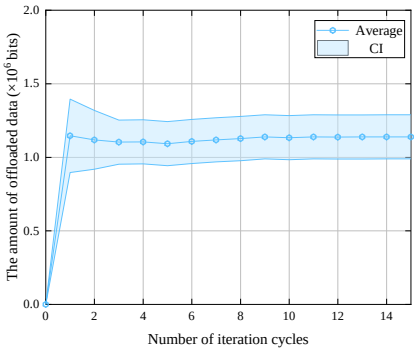


Fig. 13. Average offloaded data convergence for all users.

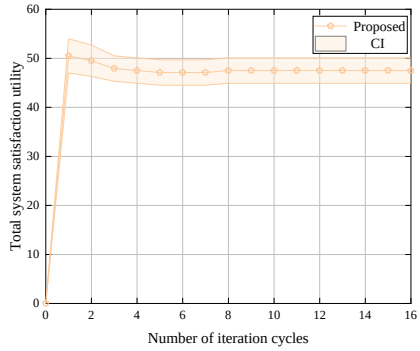


Fig. 14. Total system satisfaction utility convergence of the GTCOS algorithm.

thus the total system satisfaction utility increases. However, as the number of users further increases, the edge network resources cannot satisfy the requirements of all users, and thus the total system satisfaction utility is kept unchanged. Under our GTCOS algorithm, users can fully take advantage of edge server-based and D2D-based offloading for achieving the task offloading and resource allocation.

Since the edge server has a stable power supply from a directly connected power source, we only need to consider the system energy consumption at mobile users and idle D2D terminals. Thus, the system energy consumption is expressed as $\sum_{i\in\mathcal{N}} E_i$. Fig. 11 shows how the number of users affects system energy consumption under these four algorithms. It can be seen that the system energy consumption under each algorithm increases with the increase of the number of users. Note that compared to the other three algorithms, our GTCOS algorithm reduces the system energy consumption nearly by 10%. The reason behind these phenomena can be summarized as follows. The ACMEC and D2D schemes cannot utilize the resource allocation advantages of jointly considering the edge server and D2D-based offloading strategies, and thus a larger amount of energy is consumed for executing each task. In the UACR algorithm, the fixed resource allocation scheme allows the edge server to consume a limited amount of the system energy. The GTCOS algorithm takes into account the dynamic resource allocation such that the system consumes relatively little energy. Under our GTCOS algorithm, users can also fully take advantage of edge server-based and D2D-based offloading for conserving system energy.

Fig. 12 shows the time consumption performance of the single task under different algorithms. Since the latency requirement standard for vehicular network is no more than 50 ms, and that for online gaming is less than 100 ms, the time consumption of all algorithms shown in Fig. 12 can satisfy the requirements of latency standard. Note that the time consumption of our proposed GTCOS algorithm is a little bit higher than that of the baseline algorithms. The reason can be explained as follows. The ACMEC and D2D schemes do not execute subsequent offloading strategy updating, and only require resource allocation to accomplish the task execution. Their time consumption of task is mostly within 25 ms. The UACR algorithm integrating two offloading algorithms does not execute dynamic resource allocation. Therefore, it has a lower time consumption of task. Our proposed GTCOS algorithm adds the process of dynamic resource allocation and strategy updating, which gains a better total system satisfaction utility at the cost of sacrificing time consumption.

We then scaled up the experimental size of the edge network with 100 mobile users. Figs. 13 and 14 show the convergence (within 12 iteration cycles) of the GTCOS algorithm.

VII. Conclusion

This paper studied the computation offloading by a joint design of D2D and edge server for high-reliable and low-latency services in RCWNs. We first formulated the computation offloading as a multi-user collaborative resource dynamic management optimization problem, which fully addresses the non-uniform distribution of computational resources and the dynamic changes between computing-intensive regions and computing-sparse regions. We further proposed a dynamic offloading update strategy based on game theory to solve such an optimization problem. The simulation results show that our proposed GTCOS algorithm can achieve higher total system satisfaction utility and lower system energy consumption in comparison with the benchmark algorithms.

In future work, we will further use Markov model or real trace-based evaluation to investigate the time-varying characteristics of D2D link and task arrival rates in order to enhance the model's ability to capture load fluctuations in real-world scenarios. Then, we plan to incorporate a more rigorous model using queuing theory (e.g., M/M/1/K models) or server utilization thresholds to increase the realism and generalizability of the approach. To further improve the scalability and robustness of the system in environments with a large number of users, we propose to adopt distributed protocols or asynchronous game update mechanisms.

References

[1] F. Liu, J. Chen, Q. Zhang and B. Li, "Online MEC Offloading for V2V Networks," in IEEE Transactions on Mobile Computing, vol. 22, no. 10, pp. 6097-6109, Oct. 2023.

[2] Y. Chen, K. Li, Y. Wu, J. Huang and L. Zhao, "Energy Efficient Task Offloading and Resource Allocation in Air-Ground Integrated MEC Systems: A Distributed Online Approach," in IEEE Transactions on Mobile Computing, vol. 23, no. 8, pp. 8129-8142, Aug. 2024.

[3] W. Qi, Q. Song, L. Guo and A. Jamalipour, "Energy-Efficient Resource Allocation for UAV-Assisted Vehicular Networks With Spectrum Sharing," in IEEE Transactions on Vehicular Technology, vol. 71, no. 7, pp. 7691-7702, Jul. 2022.

[4] A. Imteaj, U. Thakker, S. Wang, J. Li and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 1-24, Jan. 2022.

[5] X. Wang et al., "Energy-Friendly Federated Neural Architecture Search for Industrial Cyber-Physical Systems," in IEEE Journal on Selected Areas in Communications, 2025.

[6] D. S. Linthicum, "Connecting Fog and Cloud Computing," in IEEE Cloud Computing, vol. 4, no. 2, pp. 18-20, Mar.-Apr. 2017.

[7] J. Tang, W. Zhao, J. Jin, Y. Xiang, X. Wang and Z. Zhou, "Adaptive Search and Collaborative Offloading Under Device-to-Device Joint Edge Computing Network," in IEEE Transactions on Mobile Computing, 2025.

[8] P. Qin, Y. Fu, X. Feng, X. Zhao, S. Wang and Z. Zhou, "Energy-Efficient Resource Allocation for Parked-Cars-Based Cellular-V2V Heterogeneous Networks," in IEEE Internet of Things Journal, vol. 9, no. 4, pp. 3046-3061, Feb. 2022.

[9] X. -Q. Pham, T. Huynh-The, E. -N. Huh and D. -S. Kim, "Partial Computation Offloading in Parked Vehicle-Assisted Multi-Access Edge Computing: A Game-Theoretic Approach," in IEEE Transactions on Vehicular Technology, vol. 71, no. 9, pp. 10220-10225, Sep. 2022.

[10] Y. Yuan, T. Yang, Y. Hu, H. Feng and B. Hu, "Two-Timescale Resource Allocation for Cooperative D2D Communication: A Matching Game Approach," in IEEE Transactions on Vehicular Technology, vol. 70, no. 1, pp. 543-557, Jan. 2021.

[11] L. Shi, L. Zhao, G. Zheng, Z. Han and Y. Ye, "Incentive Design for Cache-Enabled D2D Underlaid Cellular Networks Using Stackelberg Game," in IEEE Transactions on Vehicular Technology, vol. 68, no. 1, pp. 765-779, Jan. 2019.

[12] H. Xiang, Y. Yang, G. He, J. Huang and D. He, "Multi-Agent Deep Reinforcement Learning-Based Power Control and Resource Allocation for D2D Communications," in IEEE Wireless Communications Letters, vol. 11, no. 8, pp. 1659-1663, Aug. 2022.

[13] Y. -C. Wu, T. Q. Dinh, Y. Fu, C. Lin and T. Q. S. Quek, "A Hybrid DQN and Optimization Approach for Strategy and Resource Allocation in MEC Networks," in IEEE Transactions on Wireless Communications, vol. 20, no. 7, pp. 4282-4295, Jul. 2021.

[14] G. Zheng, C. Xu, M. Wen and X. Zhao, "Service Caching Based Aerial Cooperative Computing and Resource Allocation in Multi-UAV Enabled MEC Systems," in IEEE Transactions on Vehicular Technology, vol. 71, no. 10, pp. 10934-10947, Oct. 2022.

[15] H. Djigal, J. Xu, L. Liu and Y. Zhang, "Machine and Deep Learning for Resource Allocation in Multi-Access Edge Computing: A Survey," in IEEE Communications Surveys & Tutorials, vol. 24, no. 4, pp. 2449-2494, Fourthquarter 2022.

[16] H. Xing, L. Liu, J. Xu and A. Nallanathan, "Joint Task Assignment and Resource Allocation for D2D-Enabled Mobile-Edge Computing," in IEEE Transactions on Communications, vol. 67, no. 6, pp. 4193-4207, Jun. 2019.

[17] T. Dbouk, A. Mourad, H. Otrok, H. Tout and C. Talhi, "A Novel Ad-Hoc Mobile Edge Cloud Offering Security Services Through Intelligent Resource-Aware Offloading," in IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1665-1680, Dec. 2019.

[18] L. Pu, X. Chen, J. Xu and X. Fu, "D2D Fogging: An Energy-Efficient and Incentive-Aware Task Offloading Framework via Network-assisted D2D Collaboration," in IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3887-3901, Dec. 2016.

[19] X. Dai, Z. Xiao, H. Jiang, M. Alazab, J. Lui, S. Dustdar and J. liu, "Task Co-Offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of Things," in IEEE Transactions on Industrial Informatics, vol. 19, no. 1, pp. 480-490, Jan. 2023.

[20] P. Qin, Y. Fu, Y. Xie, K. Wu, X. Zhang and X. Zhao, "Multi-Agent Learning-Based Optimal Task Offloading and UAV Trajectory Planning for AGIN-Power IoT," in IEEE Transactions on Communications, vol. 71, no. 7, pp. 4005-4017, Jul. 2023.

[21] P. Qin, Y. Wang, Z. Cai, J. Liu, J. Li and X. Zhao, "MADRL-Based URLLC-Aware Task Offloading for Air-Ground Vehicular Cooperative Computing Network," in IEEE Transactions on Intelligent Transportation Systems, 2023.

[22] I. Ioannou, V. Vassiliou, C. Christophorou and A. Pitsillides, "Distributed Artificial Intelligence Solution for D2D Communication in 5G Networks," in IEEE Systems Journal, vol. 14, no. 3, pp. 4232-4241, Sep. 2020.

[23] H. Zhang, S. Chong, X. Zhang and N. Lin, "A Deep Reinforcement Learning Based D2D Relay Selection and Power Level Allocation in mmWave Vehicular Networks," in IEEE Wireless Communications Letters, vol. 9, no. 3, pp. 416-419, Mar. 2020.

[24] J. Du, T. Lin, C. Jiang, Q. Yang, C. F. Bader and Z. Han, "Distributed Foundation Models for Multi-Modal Learning in 6G Wireless Networks," in IEEE Wireless Communications, vol. 31, no. 3, pp. 20-30, June 2024.

[25] J. Du, C. Jiang, J. Wang, Y. Ren and M. Debbah, "Machine Learning for 6G Wireless Networks: Carrying Forward Enhanced Bandwidth, Massive Access, and Ultrareliable/Low-Latency Service," in IEEE Vehicular Technology Magazine, vol. 15, no. 4, pp. 122-134, Dec. 2020.

[26] Y. Yuan and W. Su, "A Game Theory-Based Strategy for Allocating and Offloading Computing Resources in 5G Networks," 2023 International Conference on Networking and Network Applications (NaNA), Qingdao, China, 2023, pp. 92-97.

[27] X. Wang et al., "A Resource Management Strategy for Fluid Equilibrium in Edge-Cloud Market Supporting AIGC Services," in IEEE Transactions on Services Computing, 2025.

[28] Z. Liu, Y. Zhao, J. Song, C. Qiu, X. Chen and X. Wang, "Learn to Coordinate for Computation Offloading and Resource Allocation in Edge Computing: A Rational-Based Distributed Approach," in IEEE Transactions on Network Science and Engineering, vol. 9, no. 5, pp. 3136-3151, 1 Sept.-Oct. 2022.

[29] G. Mitsis, E. E. Tsiropoulou and S. Papavassiliou, "Price and Risk Awareness for Data Offloading Decision-Making in Edge Computing Systems," in IEEE Systems Journal, vol. 16, no. 4, pp. 6546-6557, Dec. 2022.

[30] P. Vamvakas, E. E. Tsiropoulou and S. Papavassiliou, "Risk-Aware Resource Control with Flexible 5G Access Technology Interfaces," 2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Washington, DC, USA, 2019, pp. 1-9.

[31] P. A. Apostolopoulos, E. E. Tsiropoulou and S. Papavassiliou, "Risk-Aware Data Offloading in Multi-Server Multi-Access Edge Computing Environment," in IEEE/ACM Transactions on Networking, vol. 28, no. 3, pp. 1405-1418, Jun. 2020.
[32] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz and J. Rodriguez, "Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract-Matching Approach," in IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3113-3125, Apr. 2019.
[33] P. Qin, Y. Fu, G. Tang, X. Zhao and S. Geng, "Learning Based Energy Efficient Task Offloading for Vehicular Collaborative Edge Computing," in IEEE Transactions on Vehicular Technology, vol. 71, no. 8, pp. 8398-8413, Aug. 2022.
[34] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou and S. Papavassiliou, "Data Offloading in UAV-Assisted Multi-Access Edge Computing Systems Under Resource Uncertainty," in IEEE Transactions on Mobile Computing, vol. 22, no. 1, pp. 175-190, Jan. 2023.
[35] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," in Handbook of the Fundamentals of Financial Decision Making: Part I. Singapore: World Scientific, pp. 99–127, 2013.
[36] S. Boyd, L. Vandenberghe, "Convex optimization," in Cambridge university press, 2004.
[37] H. Guo and J. Liu, "Collaborative Computation Offloading for Multiaccess Edge Computing Over Fiber–Wireless Networks," in IEEE Transactions on Vehicular Technology, vol. 67, no. 5, pp. 4514-4526, May. 2018.

Yuan Yuan was born in 1997. He received the B.E. and Ph.D. degrees in information and communication engineering from National Engineering Research Center of Advanced Network Technologies, Beijing Jiaotong University, Beijing, China, in 2018 and 2025, respectively.

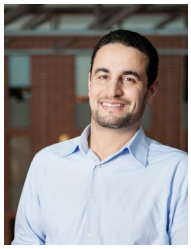
He is currently an Assistant Researcher with the Zhongguancun Laboratory, Beijing, China. He is mainly engaged in the research of the next generation Internet, vehicular edge computing and cyber security.

Bin Yang received his Ph.D. degree in systems information science from Future University Hakodate, Japan in 2015. He was a research fellow with the School of Electrical Engineering, Aalto University, Finland, from Nov. 2019 to Nov. 2021. He is currently a professor with the School of Computer and Information Engineering, Chuzhou University, China. His research interests include unmanned aerial vehicle networks, cyber security and Internet of Things.

Wei Su was born in October 1978. He got the Ph.D. degrees in Communication and Information Systems from Beijing Jiaotong University in January 2008. Now he is a teacher in the School of Electronics and Information Engineering, Beijing Jiaotong University. He granted the title of professor in November 2015. Prof. Su is mainly engaged in researching key theories and technologies for the next generation Internet and has taken part in many national projects such as National Basic Research Program (also called 973 Program), the Projects of Development Plan of the State High Technology Research, the National Natural Science Foundation of China.

Hongke Zhang (Fellow, IEEE) received M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively. He is currently a Professor at the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China, and the Director of the National Engineering Research Center for Advanced Network Technologies, Beijing, China. He is an Academician of China Engineering Academy, Beijing, China, and the Co-Director of the PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China. His current research interests include architecture and protocol design for the future Internet and specialized networks. He has authored more than ten books and is the holder of more than 70 patents. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. Prof. Zhang currently serves as an associate editor for the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and IEEE INTERNET OF THINGS JOURNAL.

Qi Liu received the B.S. degree in information and communication engineering and Ph.D. degree in communication and information system from Beijing Jiaotong University, Beijing, China, in 2003 and 2009, respectively. Then she works as a post-doctor in electronic engineering department in Tsinghua University from 2009 to 2011. She is currently a professorate senior engineer in Smart City Research Institute of China Unicom. Her research interests focus on 5G networks, cooperation of heterogeneous networks, Internet of vehicles and High-Precision Positioning.



Tarik Taleb (Senior Member, IEEE) received the B.E. degree Information Engineering with distinction and the M.Sc. and Ph.D. degrees in Information Sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Professor at the Centre for Wireless Communications (CWC) – Networks and Systems Unit, Faculty of Information Technology and Electrical Engineering, The University of Oulu. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). Between Oct. 2014 and Dec. 2021, he was a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. Before joining NEC and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. His research interests lie in the field of telco cloud, network softwarization & network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, and next generation mobile networking. He has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. He served as the general chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakech, Morocco. He was the guest editor in chief of the IEEE JSAC Series on Network Softwarization & Enablers. He was on the editorial board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE WIRELESS COMMUNICATIONS MAGAZINE, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a number of Wiley journals. Till Dec. 2016, he served as chair of the Wireless Communications Technical Committee.