Overlapping task offloading and resource allocation via multi-RSU collaborative load balancing in vehicular edge computing

Article

Accepted Version

Overlapping Task Offloading and Resource Allocation via Multi-RSU Collaborative Load Balancing in Vehicular Edge Computing

Dun Cao, *Member, IEEE,* Chi Peng, Bo Peng, Penglu Liu,
R. Simon Sherratt, *Fellow, IEEE,* Jin Wang, *Senior Member, IEEE*

***Abstract*—Vehicular Edge Computing (VEC) reduces latency by offloading vehicle generated tasks to Roadside Unit (RSU), unlocking vast opportunities for in vehicle electronics commercial services. Existing studies on task offloading in multi-RSU scenarios suffer from two major gaps. First, the similar sub-tasks across different tasks themselves have not been sufficiently investigated. This oversight leads to redundant computing, thereby undermining system efficiency. Second, and more critically, the load imbalance stemming from the uneven distribution of vehicles is further exacerbated by the neglect of similar sub-tasks. This paper focuses on redundant computation of similar sub-tasks in multi-RSU scenarios. We characterize similar sub-tasks of overlapping tasks, thereby capturing the redundancy that exists across these overlapping tasks. We then model the offloading of similar sub-tasks as a multi-objective Mixed Integer Nonlinear Program (MINLP) problem that simultaneously minimizes latency, energy consumption, and load imbalance. The original problem is approximated by a weighted-sum multi-objective problem, and we prove by contradiction that any optimal solution to this weighted-sum problem is a Pareto-optimal solution to the original multi-objective problem. To solve the constructed MINLP, we propose the Multi-RSU Distributed Shared Offloading (MRDSO) scheme. In this scheme, discrete variables are fixed based on Benders partitioning theorem, reducing the problem to a linear-programming sub-problem of the continuous variables. By proving the convexity of this sub-problem, we have demonstrated that a global optimum exists for the original MINLP problem. We then design two algorithms to solve it. Finally, experimental results confirm that this scheme reduces average system computing latency and energy consumption while maintaining multi-RSU load balance. Compared to the existing SSO, LAGO, RO, the proposed MRDSO has at least 54.8% improvement in the system cost.**

***Index Terms*—Vehicular edge computing, multi-RSU collaborative, partial offloading, shared offloading, redundant computing.**

I. Introduction

THE data produced by in vehicle electronics is growing exponentially, resulting in traditional cloud computing electronics commercial models being unable to process it in time due to excessive transmission latency [1]–[5]. Therefore, Vehicular Edge Computing (VEC) has emerged to deploy computing units such as Roadside Unit (RSU) around the vehicle [6]–[9], realizing real-time processing, then providing strong support for efficient vehicle operation and intelligent traffic management.

Nevertheless, individual RSU edge nodes are intrinsically constrained in their computing resources. Meanwhile, the uneven geographical distribution of vehicles produces RSU in hotspot regions with dense traffic flows and surging task concurrency, whereas RSU in coldspot regions remain sparsely occupied or even empty. Such inherent imbalance causes instantaneous overload in a subset of RSUs while leaving others chronically under-utilized, resulting in low global resource utilization and a severe degradation of the Quality of Service (QoS). In a local cluster of neighboring RSU, fiber-optic cable links integrated backhaul can interconnect the otherwise isolated RSUs into a unified resource pool, enabling inter-RSU collaborative offloading. In contrast to conventional isolated schemes, this edge cooperation [10]–[14] allows tasks initially offloaded to an overloaded RSU to be further migrated to an adjacent lightly-loaded RSU via low-latency wired links, thereby improving the overall performance of the VEC system through fine-grained load balancing.

However, the above studies overlook potential similarities among different vehicle applications. This omission results in the redundant computation for these similar applications. Fortunately, studies [15] has shown that such similarity often exists at the sub-task level. In practical vehicular scenarios, tasks can be decomposed into multiple parallelizable sub-tasks. Vehicles traveling along the same road segment may simultaneously process sub-tasks such as traffic signal recognition, traffic flow estimation, road condition monitoring, and environmental sensing, which are independent in execution while sharing similar data sources and computational requirements. We define such sub-tasks as similar sub-tasks. Accordingly, tasks that contain similar sub-tasks across different vehicles are defined as overlapping tasks, enabling the identification and elimination of redundant computation.

Inspired by this, to further reduce redundant computing and enhance computing efficiency in highly dynamic road traffic environments, the following critical challenges remain for overlapping task offloading in multi-RSU collaborative scenarios. First, how to quantify similar sub-tasks and characterize the overlapping relationships among tasks, thereby formulating a system-level multi-objective optimization problem. Second, whether the optimization problem admits a polynomial-time feasible solution. Finally, how to devise a low-complexity

This work was supported in part by the National Natural Science Foundation of China (NSFC) (Grant No.62572076, Grant No. 62272063, Grant No. 62473146 and Grant No.U25A20426), the Hunan Provincial Natural Science Foundation of China (Grant No. 2025JJ20069), the Research Foundation of Education Bureau of Hunan Province under (Grant No. 23A0253), the Key Project of Natural Science Foundation of Hunan Province under Grant (Grant No. 2024JJ3017).

Dun Cao, Chi Peng, Bo Peng, and Penglu Liu are with Changsha University of Science and Technology, Changsha, Hunan 410076, China. (e-mail: caodun@csust.edu.cn, 23208031767@stu.csust.edu.cn, 1299511163@qq.com, LPL@csust.edu.cn).

R. Simon Sherratt is with the School of Biomedical Engineering, University of Reading, RG6 6AH Reading, U.K. (e-mail: r.s.sherratt@reading.ac.uk).

Jin Wang is a Professor in the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China. (e-mail: jinwang@hnust.edu.cn).

algorithm that efficiently solves the intricate problem coupling latency, energy consumption, and load balancing.

To summarize, our paper's contributions can be stated as follows in comparison to previous research:

- To accurately capture the overlapping relationships among tasks in a multi-RSU shared offloading scenario, we take the similar sub-task as the finest modeling granularity. Concurrently, we comprehensively model the shared offloading computation paradigm for similar sub-tasks under diverse scenarios, devise a load balancing metric, and formulate a multi-objective optimization problem that jointly minimizes latency, energy consumption, and load imbalance. By means of proof by contradiction, we demonstrate that the problem can be approximated by a weighted-sum multi-objective problem.
- Given that the formulated optimization is a Mixed Integer Nonlinear Program (MINLP) containing both continuous and discrete variables, we apply Benders partitioning theorem to fix the discrete variables to convert the problem into a linear programming subproblem in the continuous variables. By exploiting the polyhedral cone properties of this subproblem, we analyze its relationship to the original optimum and formally prove that the MINLP can be decoupled into two sequentially coupled subproblems, one for the discrete variables and one for the continuous variables.
- Given the complexity of the problem and its constraints, we define a repetition weight and design a targeted solution scheme based on it. We propose a Multi-RSU Distributed Shared Offloading (MRDSO) scheme consisting of two key phases. First, a Two Layer Collaborative Sharing (TLCS) algorithm is employed to derive the optimal offloading policy and the execution sequence of sub-tasks. Second, a resource allocation based on the Genetic Algorithm (GA) is designed to determine the proportion of computing resources that each RSU should devote to different types of similar sub-tasks.

II. Relate work

In this section, we conduct a comprehensive and systematic analysis of the existing works concerning sub-task offloading in multi-RSU enabled vehicular edge computing environments.

A. Task offloading for sub-task

Binary offloading, as adopted in the previous studies, is simple and direct. However, it lacks flexibility as tasks are either fully executed locally or entirely offloaded to the edge, preventing the system from selecting an appropriate offloading strategy based on sub-tasks characteristics, resource availability, and network conditions.

Consequently, some studies have begun to focus on partial offloading, which allows tasks to be divided as needed and enables flexible determination of the local and offloading execution portions. L. Zhao [16] proposed a digital twin-assisted partial offloading strategy for vehicle tasks. By performing parallel computing of splittable vehicle tasks on both vehicles and RSUs, this strategy effectively reduces task latency. H. Zhou [17] studied the partial offloading of Deep Neural Network (DNN) in MEC, aiming to accelerate DNN inference through model parallelism and partial offloading. L. Zhao [18] proposes a Lyapunov-guided SAC-based method to jointly optimize delay, energy consumption, and queue stability in ISAC-aided IoV environments. H. Lin [19] builds a Stackelberg game where RSUs lead and vehicles follow, balancing task latency, vehicle energy consumption and RSU residual resources to motivate RSUs to help with utility maximizing partial offloading. S. Lei [20] proposed a federated MADDPG-based collaborative scheduling framework to jointly optimize task offloading and resource allocation in a dynamic multi-RSU environment. The above studies adopt partial offloading, which is more efficient, flexible and reliable than binary offloading, and can optimize resource utilization and task execution. However, these studies overlook the relationships between tasks and instead model tasks as arbitrarily divisible segments. This approach is flawed because tasks are typically composed of multiple sub-tasks. As the smallest unit of measurement, sub-tasks should form the basis for offloading decisions. Furthermore, due to similarities across different scenarios, tasks generated by multiple vehicles at the same moment may contain overlapping sub-tasks, such as traffic light analysis and processing. Although these sub-tasks belong to different overlapping tasks, the computational data, programs, and execution logic they require are identical. Therefore, effectively eliminating redundant computing of overlapping tasks is imperative, a challenge that traditional partial offloading research struggles to address.

B. Task offloading for multi-RSU scenarios

As a key node in the edge computing network, the selection of RSUs has an important impact on the efficiency and effectiveness of task offloading. Reasonable RSU selection in multi-RSU scenarios can ensure that tasks are offloaded to the most appropriate service nodes, thus achieving efficient resource utilization and fast task processing. X. Zhu [21] consider multiple vehicles selecting one of multiple MEC servers in the vicinity so that the vehicles can make the best offloading decisions to minimise task processing latency.

Due to the mobility of vehicles, their positions and speeds are constantly changing, which results in the communication links between vehicles and RSUs as well as other vehicles changing dynamically over time. When a vehicle moves, it may go beyond the communication coverage of the RSU or other vehicles, resulting in interruption or loss of the communication link. W. Zhao [22] takes into account that each task can only be offloaded to one node, that the task needs to be completed in a constrained time, and that it prevents the failure of task processing due to communication interruptions. In addition, a forwarding vehicle selection scheme is designed to identify vehicles that can communicate stably with the source RSU and determine the optimal communication path for successful task forwarding. S. Li [23] proposes two offloading options. One is to offload the task to RSUs that the vehicle may pass through, which requires the vehicle to drive into the coverage area of the selected RSU for offloading.

TABLE I
COMPARISON OF REPRESENTATIVE VEC OFFLOADING SCHEMES

Scheme	Redundancy-aware	Multi-RSU	Optimization Objective			Resource Allocation	Sub-task Order Optimized
			Delay	Energy	Load Balancing		
Ours	Yes	Yes	Yes	Yes	Yes	Yes	Yes
[16]	No	Yes	Yes	No	No	Yes	No
[17]	No	No	Yes	No	No	Yes	Yes
[18]	No	No	Yes	Yes	No	Yes	No
[19]	No	No	Yes	Yes	Yes	Yes	No
[20]	No	Yes	Yes	No	Yes	Yes	No
[21]	No	Yes	Yes	No	No	No	No
[22]	No	Yes	Yes	No	No	No	No
[23]	No	Yes	Yes	No	No	Yes	No
[24]	No	Yes	Yes	No	Yes	Yes	No
[25]	No	Yes	Yes	No	No	Yes	No

The other option is to offload the task to an RSU that the vehicle will reach through radio communication between neighbouring RSUs, which is limited by radio resources and possible multi-hop transmissions. The main focus of the study is on optimizing the total processing latency of a task through multi-hop task offloading and deep reinforcement learning methods. However, these studies does not make full use of the collaboration between RSUs.

Subsequently, Y. Cui [24] balance RSU loads through an adaptive cooperative offloading algorithm driven by branch-and-bound iteration. Z. Zhang [25] boost RSU utilization by edge-cooperatively shifting tasks from congested to idle servers. These studies enhance system computing efficiency and balance RSU load degree by optimizing initial task offloading and secondary task offloading in multi-RSU system. However, none of the aforementioned studies address the critical fact that overlapping tasks exist in multi-RSU system. They overlook the repeated computing of similar sub-tasks, which ultimately become heavy anchors that slow down the overall system efficiency.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System model

As illustrated in Fig. 1, tasks generated by vehicles may consist of multiple similar sub-tasks. For each type of similar sub-task, one vehicle is selected as the offloading node, while other vehicles with the same sub-task obtain computation results through V2V or R2V communication. Vehicles marked with the same color correspond to the selected offloading nodes for each sub-task type. As depicted in Fig. 1, VEC architecture comprises M RSUs, each fitted with a MEC server boasting computing resources of F_m^{rsu}, and vehicles in their communication ranges. In which we denote the set of all RSUs as $\mathcal{R} = \{R_1, ..., R_m, ..., R_M\}, m \in [1, M]$, the location of R_m is defined as $loc_m = \{x_m, y_m\}$, where x_m and y_m are the x and y coordinates of RSU R_m. Each RSU has a communication range of R_{rsu} and is situated at various intersections and road segments. In our study, vehicle that initiate tasks are termed task vehicle (TV). All of these TVs can issue task requests simultaneously, with each task being further divided into independent sub-tasks. Our scenario includes N vehicles, represented by the set $\mathcal{V} = \{v_1, ..., v_n, ..., v_N\}, n \in [1, N]$.

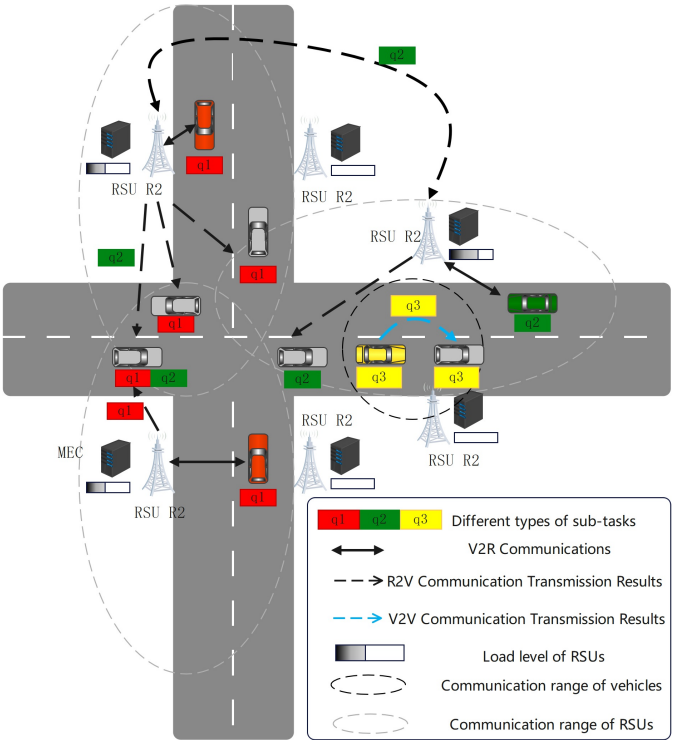

Fig. 1. The multi-RSU system scenarios.

Each vehicle is equipped with computing resource of f_n and has a communication range of R_{veh}. Each vehicle in the system has a randomly distributed initial position and carries a BeiDou Navigation Satellite System (BDS). For any vehicle $v_n \in \mathcal{V}$, its location at time t is defined as $loc_n^t = \{x_n^t, y_n^t\}$, where x_n^t and y_n^t are the x and y coordinates of vehicle v_n at time t. In this paper, the main symbols used are shown in TABLE II.

B. Task model

In this system, without loss of generality, we assume that each TV generates one task per time slot. A task is modeled as $S_n = \{T_n^{\max}, \mathbf{M}_n\}$, where $T_n^{\max}$represents the maximum tolerable latency for task S_n, and $\mathbf{M}_n$ denotes the set of sub-tasks for task S_n. When v_n generates a task, it comprises

TABLE II
SYMBOL AND DEFINITION.

Symbol	Definition
N	Number of vehicles
K	Number of sub-tasks types
M	Number of RSUs
$\mathcal{V}$	Set of TVs
$\mathcal{R}$	Set of RSUs
$\mathcal{B}$	Set of number of sub-tasks
v_n	The n-th vehicle
$T_n^{\max}$	Latency constraint of task S_n
$E_n^{\max}$	Energy constraint of each vehicle v_n
$\mathbf{M}_n$	Set of sub-tasks of task S_n
q_k	The k-th type of sub-task
$q_{n,k}$	The k-th type of sub-task generated by v_n
f_n	Computing capacity of vehicle v_n
F_m^{rsu}	Computing capacity of RSU R_m
$r_{n,i}$	Uplink rate between vehicle v_n and v_i
B^{V2R}	Bandwidth of V2R communication
B^{V2V}	Bandwidth of V2V communication
p_1	Fading factor of the link channel
p_n	Transmisson power
ρ_0	Gaussian noise density
ε	Energy consumption parameters
β_1	Weighting coefficient for latency
β_2	Weighting coefficient for energy consumption
β_3	Weighting coefficient for load balancing
γ	Energy proportion factor
$\mathcal{X}$	Set of offloading strategy
$\mathcal{P}$	Set of sub-task computing sequences on vehicles
$\mathcal{F}$	Set of resource allocation
R_{rsu}	Communication range of the RSU
R_{veh}	Communication wrange of vehicle v_n

$||\mathbf{M}_n||$ sub-tasks. Each sub-task $q_{n,k}$ is modeled as a tuple $q_{n,k} = \{d_k, c_k, o_k\}, k \in [1, K]$, where d_k is the data size of sub-task $q_{n,k}$, c_k is the total Central Processing Unit (CPU) cycles required, and o_k is the size of the result data. These sub-tasks are recorded in the set $\mathbf{M}_n = \{q_{n,k}\}$. For simplicity, we denote sub-task q_k without the vehicle index when the similar sub-tasks is generated by different TVs, and use the set $\mathbf{W}_k$ to represent the TVs generating sub-task q_k. There are K types of sub-tasks in the system, which are included in the set $\mathcal{K} = \{q_1, q_2, ..., q_k, ..., q_K\}, k \in [1, K]$. We count the number of requests for each type of sub-tasks in the system as $\mathcal{B} = \{b_1, b_2, ..., b_k, ..., b_K\}, k \in [1, K]$, where each b_k represents the number of the k-th type of sub-task q_k.

C. Communication model

In a multi RSU scenario, Roadside-to-Roadside (R2R) communication relies on fiber-optic links with uniform transmission rates [26], [27]. For Vehicle-to-Roadside (V2R), Roadside-to-Vehicle (R2V) and Vehicle-to-Vehicle (V2V) communication, the Shannon formula [28] is used to express the task data transmission rate as follow:

$$r = B\log_2\left(1 + \frac{p_1^2\left(\frac{1}{l}\right)^{p_2} p_n}{B\rho_0}\right), \tag{1}$$

where $r = \{r_{n,m}, r_{m,n}, r_{n,j}\}$ denote the transmission rates from vehicle v_n to RSU R_m, RSU R_m to vehicle v_n, and vehicle v_n to vehicle v_j, respectively. $B = \{B^{\mathrm{V2R}}, B^{\mathrm{R2V}}, B^{\mathrm{V2V}}\}$ denote the transmission bandwidth for V2R, R2V and V2V communication, respectively, p_1 represents the channel gain, p_2 represents the path loss exponent. $l = \{l_{n,m}^t, l_{m,n}^t, l_{n,j}^t\}$ denotes the Euclidean distance from vehicle v_n to R_m, and that $l_{n,m}^t$, R_m to vehicle v_n, and vehicle v_n to vehicle v_j, and $l_{m,n}^t$ have the same value, ρ_0 represents the Gaussian noise density during communication, and $p_n = \{p_n^{\mathrm{V2V}}, p_n^{\mathrm{V2R}}\}$ denotes the transmission power of V2V and V2R.

The distance between any two V2V nodes and V2R nodes at time t can be calculated using the L2-norm and is denoted as follow:

$$\begin{aligned} l_{n,i}^t =& \left\|loc_n^t, loc_i^t\right\|_2 = \sqrt{\left(x_n^t - x_i^t\right)^2 + \left(y_n^t - y_i^t\right)^2}, \\ & v_n \in \mathcal{V}, v_i \in \mathcal{V}, n \neq i. \end{aligned} \tag{2}$$

$$\begin{aligned} l_{n,m}^t =& \left\|loc_n^t, loc_m\right\|_2 = \sqrt{\left(x_n^t - x_m\right)^2 + \left(y_n^t - y_m\right)^2}, \\ & v_n \in \mathcal{V}, R_m \in \mathcal{R}. \end{aligned} \tag{3}$$

D. Computing model

For each independent sub-task q_k on a TV, determining its offloading position is essential. We introduce the offloading strategy set $\mathcal{X}_{n,k} = \left\{x_{n,k}^{\mathrm{loc}}, x_{n,k,m}^{\mathrm{rsu}}, x_{n,k,j}^{\mathrm{shared}}\right\}, n, j \in [1, N], k \in [1, K], m \in [1, M], n \neq j$, where binary variable $x_{n,k}^* \in \{0, 1\}$ shows if sub-task q_k will computing at a specific node, where 1 means yes and 0 means no. The offloading strategy sets for all TVs in the system are represented as $\mathbf{X} = \{\mathcal{X}_{n,k}\}, v_n \in \mathcal{V}, q_k \in \mathbf{M}_n$. Similarly, $x_{n,k,m}^{\mathrm{rsu}}$ indicates whether sub-task q_k of vehicle v_n is offloading to the RSU R_m, $x_{n,k,j}^{\mathrm{shared}}$ indicates whether vehicle v_n obtains the computing result of sub-task q_k for vehicle v_j through shared offloading, where the optimal execution method of vehicle v_j for sub-task q_k can be either local computing or offloading to the RSU.

In VEC, considering the heterogeneity of vehicles, in order to achieve the goal of minimizing the latency and energy consumption of the system tasks in the state of load balancing of the computing units, the optimal computing nodes need to be selected for the tasks. Based on the differences in task processing nodes and computing methods, we classify the task processing methods into local computing, edge computing and shared offloading. In the following section, we systematically analyze the sub-task computing model based on these three methods.

1) Local computing for sub-task: For local computing, the computing latency of sub-task q_k on vehicle v_n is given as follow:

$$t_{n,k}^{\mathrm{loc}} = \frac{c_k}{f_n}. \tag{4}$$

The energy consumption for local computing is given as follow:

$$e_{n,k}^{\mathrm{loc}} = \varepsilon(f_n)^2 c_k, \tag{5}$$

where c_k represents the number of CPU cycles required to complete sub-task q_k, and f_n denotes the computing capacity of vehicle v_n. According to the Dynamic Voltage and Frequency Scaling (DVFS) technique, the CPU power consumption model is given by $P = \varepsilon f^{\phi+1}$ [29], ε is a coefficient determined by the chip architecture, and $\phi > 1$ is

the energy consumption coefficient. The power consumption per cycle can be calculated as εf^{ϕ}, since f corresponds to the number of cycles per second. For the sake of computing simplicity, ϕ is set to 2 [30].

Given that vehicles have significantly lower computing resource than RSU equipped with MEC servers, we assume tasks on vehicles are executed through serial computing. As vehicles conduct serial computing, if sub-task $q_{k'}$ is already being computing on vehicle v_n and sub-task q_k also needs computing there, q_k must queue until v_n is available. $AT_{n,k}$ represents the earliest activate time when vehicle v_n can start computing sub-task q_k. The earliest finish time of q_k is the sum of its queuing time and computing latency on v_n. Thus, the earliest finish time $EFT_{n,k}$ for sub-task q_k on vehicle v_n can be calculated as follow:

$$EFT_{n,k} = AT_{n,k} + t^{\mathrm{loc}}_{n,k}, \tag{6}$$

where $t^{\mathrm{loc}}_{n,k}$ represents the local computing latency of sub-task q_k.

To determine the earliest activate time $AT_{n,k}$, we use $p_{n,k}$ to indicate the computing sequences of sub-task q_k on vehicle v_n. The calculation is as follow:

$$p_{n,k} = \sum_{\zeta_k < \zeta_{k'}} x^{\mathrm{loc}}_{n,k'}, \tag{7}$$

where $\zeta_k < \zeta_{k'}$ indicates that the repetition weight of sub-task $q_{k'}$ is higher than that of sub-task q_k. Thus, $AT_{n,k}$ is determined as follow:

$$AT_{n,k} = \begin{cases} 0, p_{n,k} = 0, \\ EFT_{n.k'} + t_{n,k'}, p_{n,k} \neq 0, \\ q_k, q_{k'} \in \mathcal{K}, p_{n,k'} = p_{n,k} - 1, \end{cases} \tag{8}$$

where $p_{n,k}$ denotes the execution sequence number of sub-task q_k on vehicle v_n, $p_{n,k} = 0$ means that sub-task q_k is the first one to be computing on vehicle v_n. $p_{n,k'} = p_{n,k} - 1$ indicates that sub-task $q_{k'}$ is the predecessor of q_k, with $EFT_{n,k'}$ being the earliest finish time of $q_{k'}$. $P_n = \{p_{n,k}\}$ represents the computing sequence for the sub-tasks on vehicle v_n, and the set $\mathcal{P} = \{P_1, P_2, ..., P_n, ..., P_N\}$ denotes the computing sequences for all TVs.

2) Edge computing for sub-task: For VEC, F^{rsu}_m represents the total computing resources of RSU R_m. As RSUs have abundant computing resources compared to vehicles, they can handle tasks simultaneously. The computing latency for offloading sub-task q_k of vehicle v_n to RSU R_m is given as follow:

$$t^{\mathrm{rsu}}_{n,k,m} = \frac{d_k}{r_{n,m}} + \frac{c_k}{F^{\mathrm{rsu}}_{m,k}} + \frac{o_k}{r_{n,m}}, \tag{9}$$

where the first part represents the transmission latency of sub-task q_k, the second part is the computing latency of sub-task q_k on RSU R_m, and the third part is the result transmission latency of sub-task q_k. $F^{\mathrm{rsu}}_{m,k}$ denotes the computing resources allocated by RSU R_m to sub-task q_k.

The energy consumption for vehicle v_n to offload sub-task q_k to RSU R_m for computing is given as follow:

$$e^{\mathrm{rsu}}_{n,k,m} = p_n \frac{d_k}{r_{n,m}} + \varepsilon \left(F^{\mathrm{rsu}}_{m,k}\right)^2 c_k + p_n \frac{o_k}{r_{m,n}}, \tag{10}$$

where the first part represents the transmission energy consumption of sub-task q_k, the second part is the computing energy consumption of sub-task q_k on RSU R_m, and the third part is the result transmission energy consumption of sub-task q_k, p_n denotes the transmission power of the vehicle.

3) Shared offloading for sub-task: Shared offloading allows vehicle v_n to reuse the computing results of vehicle v_j about sub-task q_k. It's worth noting that v_j may either complete this computation locally or offload it to RSU. Therefore, we will discuss shared offloading in different scenarios.

First, we set $\mu = 1$ when vehicle v_j computes sub-task q_k locally. The modeling decomposition of latency and energy consumption is as follows.

- When vehicle v_n is in the communication range of vehicle v_j, vehicle v_n can directly obtain the computing results via the V2V link. The latency for sharing and computing the sub-task q_k via V2V is given as follow:

$$t^{\mathrm{V2V}}_{n,k,j} = t^{\mathrm{loc}}_{j,k} + \frac{o_k}{r_{j,n}}, v_n \neq v_j, \tag{11}$$

  where the first part represents the computing latency of sub-task q_k on vehicle v_j, and the second part is the result transmission latency of sub-task q_k.
  The energy consumption required for V2V-based shared offloading of sub-task q_k is expressed as follow:

$$e^{\mathrm{V2V}}_{n,k,j} = p_n \frac{o_k}{r_{j,n}}. \tag{12}$$

- When vehicle v_n is out of communication range of vehicle v_j, RSU relaying can be employed. Vehicle v_j uploads the computation results of sub-task q_k to its associated RSU $R_{m'}$, which forwards them via a wired backbone link to the RSU R_m associated with vehicle v_n. Vehicle v_n then retrieves the results from that RSU R_m, enabling cross-range data sharing. Considering that RSUs are interconnected via fiber-optic cable links and computing results are relatively small in size, we employs a fixed value t^{R2R} to determine the result return latency between RSUs. Therefore, the latency calculation in this scenario is as follows:

$$t^{\mathrm{V2V}}_{n,k,j} = t^{\mathrm{loc}}_{j,k} + \frac{o_k}{r_{j,m'}} + t^{R2R} + \frac{o_k}{r_{m,n}}, v_n \neq v_j, \tag{13}$$

  where the first part represents the computing latency of sub-task q_k on vehicle v_j, the second part is the upload latency for results, the third part is the link transmission latency between RSUs, and the final part is the download latency for results.
  The energy consumption is expressed as follows:

$$e^{\mathrm{V2V}}_{n,k,j} = p_n \frac{o_k}{r_{m,n}}. \tag{14}$$

Second, we set $\mu = 0$ when vehicle v_j offloads sub-task q_k to the RSU $R_{m'}$ for computing. In this scenario, we initialize $\xi_{j,k} = 0$. The modeling decomposition of latency and energy consumption is as follows.

- When vehicle v_j offloads sub-task q_k at RSU $R_{m'}$, and vehicle v_n is covered by RSU R_m, and $m^{'} = m$, vehicle

v_n can directly obtain the results from the RSU R_m. The latency for R2V-based shared offloading of sub-task q_k is given as follow:

$$t_{n,k,j}^{\mathrm{R2V}} = \frac{d_k}{r_{j,m'}} + \frac{c_k}{F_{m',k}^{\mathrm{rsu}}} + \frac{o_k}{r_{m,n}}, \tag{15}$$

where the first part represents the transmission latency for vehicle v_j to offload sub-task q_k to the RSU R_m, the second part represents the computing latency of sub-task q_k on RSU R_m, and the final part is the result transmission latency of sub-task q_k.

Since shared offloading eliminates the overhead of vehicle v_n uploading data for sub-task q_k, energy consumption only needs to account for the transmission of computing results. The energy consumption for V2R-based shared offloading of sub-task q_k is given as follow:

$$e_{n,k,j}^{\mathrm{R2V}} = p_n \frac{o_k}{r_{n,m}}. \tag{16}$$

- When vehicle v_j offloads sub-task q_k at RSU $R_{m'}$, and vehicle v_n is covered by RSU R_m, and $m^{'} \neq m$, vehicle v_n can obtain results via fiber-optic cable links between RSUs. In this scenario, we set $\xi_{j,k} = 1$. The latency is as follows:

$$t_{n,k,j}^{\mathrm{R2V}} = \frac{d_k}{r_{j,m'}} + \frac{c_k}{F_{m',k}^{\mathrm{rsu}}} + t^{R2R} + \frac{o_k}{r_{m,n}}, \tag{17}$$

where the first part represents the transmission latency for vehicle v_j to offload sub-task q_k to the RSU $R_{m'}$, the second part is the computing latency of sub-task q_k on RSU $R_{m'}$, the third part is the link transmission latency between RSUs, and the final part is the download latency for results.

The energy consumption is expressed as follows:

$$e_{n,k,j}^{\mathrm{R2V}} = p_n \frac{o_k}{r_{m,n}}. \tag{18}$$

4) Computing model for task: Accordingly, the computing latency of sub-task q_k for vehicle v_n is given as follow:

$$\begin{aligned} t_{n,k} =& x_{n,k}^{\mathrm{loc}} EFT_{n,k} + x_{n,k,m}^{\mathrm{rsu}} t_{n,k,m}^{\mathrm{rsu}} \\ &+ x_{n,k,j}^{\mathrm{shared}} \left(\mu t_{n,k,j}^{\mathrm{V2V}} + (1-\mu)\, t_{n,k,j}^{\mathrm{R2V}}\right). \end{aligned} \tag{19}$$

In addition, to ensure the sub-task q_k of vehicle v_n is only processed in one place, the condition of binary variable must be satisfied as follow:

$$x_{n,k}^{\mathrm{loc}} + x_{n,k,m}^{\mathrm{rsu}} + x_{n,k,j}^{\mathrm{shared}} = 1, v_n, v_j \in \mathcal{V}, q_k \in \mathbf{M}_n. \tag{20}$$

In summary, the computing latency of the task S_n generated by vehicle v_n is given as follow:

$$T_n^{\mathrm{total}} = \max_{q_k \in \mathbf{M}_n} t_{n,k}. \tag{21}$$

The energy consumption of the task S_n generated by vehicle v_n is given as follow:

$$\begin{aligned} E_n^{\mathrm{total}} =& \sum_{q_k \in \mathbf{M}_n, v_n \in \mathcal{V}} x_{n,k}^{\mathrm{loc}} e_{n,k}^{\mathrm{loc}} + x_{n,k,m}^{\mathrm{rsu}} e_{n,k,m}^{\mathrm{rsu}} \\ &+ x_{n,k,j}^{\mathrm{shared}} \left(\mu e_{n,k,j}^{\mathrm{V2V}} + (1-\mu)\, e_{n,k,j}^{\mathrm{R2V}}\right) \\ &+ \sum_{v_j \in \mathcal{V}} \sum_{q_k \in \mathbf{M}_n} \xi_{j,k} p_n \frac{o_k}{r_{j,m'}}. \end{aligned} \tag{22}$$

where the final part represents the single-time energy consumption when the sharing party uploads results during shared offloading, requiring calculation only once.

E. Load balancing model

This paper primarily focuses on the utilization rate of RSUs' computing resources, and the load balancing metric defined as follow:

$$U = \frac{1}{M} \sum_{R_m \in \mathcal{R}} (I_m - I^{\mathrm{avg}})^2, \tag{23}$$

where I_m is the resource utilization rate of RSU R_m, and I^{avg} is the average utilization rate of all RSUs, and a smaller U indicates better load balancing among RSUs. The formulas for calculating I_m and I^{avg} are as follows:

$$I_m = \frac{1}{F_m^{\mathrm{rsu}}} \sum_{q_k \in \mathcal{K}} x_{n,k,m}^{\mathrm{rsu}} F_{m,k}^{\mathrm{rsu}}, \tag{24}$$

$$I^{\mathrm{avg}} = \frac{1}{M} \sum_{R_m \in \mathcal{R}} I_m, \tag{25}$$

where F_m^{rsu} represents all the computing resources owned by RSU R_m.

F. Problem formulation

Our objective is to minimize the average computing latency and energy consumption of all TVs while addressing the uneven load among RSUs. This involves optimizing offloading strategy, the computing sequence of sub-tasks on vehicles, and resource allocation on MEC servers. We define the utility function as follow:

$$\begin{aligned} &\min_{\mathcal{X},\mathcal{P},\mathcal{F}} Q = \min_{\mathcal{X},\mathcal{P},\mathcal{F}} \left(\sum_{v_n \in \mathcal{V}} T_n^{\mathrm{total}}, \sum_{v_n \in \mathcal{V}} \gamma E_n^{\mathrm{total}}, U \right), \\ &s.t. \mathrm{C1}: T_n^{\mathrm{total}} \leq T_n^{\max}, \forall v_n \in \mathcal{N}, \\ &\quad \mathrm{C2}: E_n^{\mathrm{total}} \leq E_n^{\max}, \forall v_n \in \mathcal{N}, \\ &\quad \mathrm{C3}: x_{n,k}^{\mathrm{loc}} + x_{n,k,m}^{\mathrm{rsu}} + x_{n,k,j}^{\mathrm{shared}} = 1, v_n, v_j \in \mathcal{V}, q_k \in \mathbf{M}_n, \\ &\quad \mathrm{C4}: x_{n,k}^{\mathrm{loc}}, x_{n,k,m}^{\mathrm{rsu}}, x_{n,k,j}^{\mathrm{shared}} \in \{0,1\}, v_n, v_j \in \mathcal{V}, q_k \in \mathbf{M}_n, \\ &\quad \mathrm{C5}: \sum_{q_k \in \mathcal{K}} F_{m,k}^{\mathrm{rsu}} \leq F_m^{\mathrm{rsu}}, \forall R_m \in \mathcal{R}, \\ &\quad \mathrm{C6}: F_m^{\min} \leq F_{m,k}^{\mathrm{rsu}} \leq F_m^{\mathrm{rsu}}, \forall R_m \in \mathcal{R}, q_k \in \mathbf{M}_n, \\ &\quad \mathrm{C7}: p_{n,k'} = p_{n,k} - 1, \\ &\qquad p_{n,k'}, p_{n,k} \in \mathbb{N}, q_k, q_{k'} \in \mathcal{K}, \forall v_n \in \mathcal{N}, \end{aligned} \tag{26}$$

where $\mathcal{X}$ represents the offloading strategy for all vehicles, $\mathcal{P}$ is the computing sequences on local vehicles, and $\mathcal{F} = \{F_{m,k}^{\mathrm{rsu}} | q_k \in \mathcal{K}, R_m \in \mathcal{R}\}$ the CPU resources allocated by RSUs. The objective function Q aims to minimize the computing latency and energy consumption of all vehicles in the system while minimizing load difference among RSUs. Constraint $C1$ sets the maximum allowable latency for task execution. Constraint $C2$ defines $E_n^{\max}$ as the energy consumed when task n is processed at RSU R_0, representing the maximum energy constraint for tasks. Constraint $C3$ ensures that each sub-tasks on a vehicle is computed at only one location. Constraint $C4$ requires offloading strategy to be binary variables. Constraint $C5$ ensures the total allocated CPU resources at each RSU do not exceed its maximum capacity.

Constraint $C6$ represent the sequential relationship between any sub-task and its successor sub-task in the computing sequence of all TVs.

In order to avoid finding an infinite Pareto-optimal solution and to ensure fairness in the consideration of the three optimization objectives [31]. Thus, as follow, our objective function can then be formulated as the weighted sum cost of optimizing the latency cost and energy cost as well as the load level cost, and this weighted sum cost is used as a metric in the results and discussion section.

$$\begin{aligned}&\min_{\mathcal{X},\mathcal{P},\mathcal{F}} Q = \min_{\mathcal{X},\mathcal{P},\mathcal{F}} \sum_{v_n\in\mathcal{V}} \left(\beta_1 T_n^{\text{total}} + \beta_2\gamma E_n^{\text{total}}\right) + \beta_3 U,\\ &s.t. \text{C1, C2, C3, C4, C5, C6, C7,}\end{aligned} \tag{27}$$

where $\beta_1, \beta_2, \beta_3 \in [0, 1]$ are the weighted coefficients for latency, energy consumption, and load balancing, respectively, with $\beta_1 + \beta_2 + \beta_3 = 1$.

Theorem 1: Any optimal solution to the optimization problem (27) is a Pareto optimal solution to the optimization problem (26).

Proof by Contradiction 1: Denote the optimal solution of the optimization problem (27) as $(\mathcal{X}^*, \mathcal{P}^*, \mathcal{F}^*)$. Assume that $(\mathcal{X}^*, \mathcal{P}^*, \mathcal{F}^*)$ is not a Pareto optimal solution to the optimization problem (26) . There must then exist another solution $(\mathcal{X}', \mathcal{P}', \mathcal{F}')$ that achieves a lower value in at least one objective and has no higher value with respect to any objective.

That is, $\min_{\mathcal{X}',\mathcal{P}',\mathcal{F}'} \sum_{v_n\in\mathcal{V}} \left(\beta_1 T_n^{\text{total}} + \beta_2\gamma E_n^{\text{total}}\right) + \beta_3 U <$ $\min_{\mathcal{X}^*,\mathcal{P}^*,\mathcal{F}^*} \sum_{v_n\in\mathcal{V}} \left(\beta_1 T_n^{\text{total}} + \beta_2\gamma E_n^{\text{total}}\right) + \beta_3 U$ case exists, contradicting Theorem 1. Therefore, Theorem 1 is proved.

In summary, the optimization problem (26) has an optimal solution. To characterize the computational complexity of problem (27), we establish its NP-hardness via a reduction argument. The problem is a Mixed Integer Nonlinear Programming (MINLP) formulation involving discrete variables $\mathcal{X}$ and $\mathcal{P}$, continuous resource allocation variables $\mathcal{F}$, and nonlinear objective and constraint functions, where latency is inversely proportional to computing frequency and energy consumption is proportional to the square of the frequency. To formally prove its NP-hardness, we consider a simplified version under the following restrictions, where each task contains only a single sub-task (i.e., $|M_n| = 1$), shared offloading is disabled (i.e., $x_{n,k,j}^{\text{shared}} = 0$), the resource allocation variables $F_{m,k}^{\text{rsu}}$ are fixed constants, and task execution sequencing is omitted. Under these assumptions, the problem reduces to a task assignment problem under RSU capacity constraints, where each task is either executed locally or assigned to one RSU. In this setting, tasks and RSUs can be mapped to items and knapsacks, respectively, and the constraint $\sum_{q_k} F_{m,k}^{\text{rsu}} \le F_m^{\text{rsu}}$ corresponds to the knapsack capacity constraint, while the objective corresponds to minimizing assignment costs. Therefore, the reduced problem can be transformed in polynomial time into the classical Generalized Assignment Problem (GAP), which is known to be NP-hard [32]. Since problem (27) is a strict generalization of this reduced formulation with additional decision variables, including shared offloading, task sequencing, and continuous resource allocation, it follows that problem (27) is also NP-hard.

Motivated by this observation, we adopt Benders partitioning to decompose the original MINLP into more tractable subproblems. Specifically, it can be observed that when the discrete variables $\mathcal{X}$ and $\mathcal{P}$ are fixed, problem (27) reduces to an optimization problem that depends solely on the continuous variable $\mathcal{F}$. For ease of exposition, the resulting subproblem with respect to $\mathcal{F}$ is defined as follows:

$$\begin{aligned}&\min_{\mathcal{F}} Q = \min_{\mathcal{F}} \sum_{v_n\in\mathcal{V}} \left(\beta_1 T_n^{\text{total}} + \beta_2\gamma E_n^{\text{total}}\right) + \beta_3 U,\\ &s.t. \text{C1, C2, C5, C6.}\end{aligned} \tag{28}$$

The second-order partial derivatives of the computing resources of a task on an RSU are related only to the computing resources on that RSU and not to the computing resources on other RSUs. Thus for each RSU there is Hessian matrix as follow:

$$H(Q) = \begin{pmatrix} \frac{\partial^2 Q}{\partial {F_{m,1}^{\text{rsu}}}^2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial^2 Q}{\partial {F_{m,K}^{\text{rsu}}}^2} \end{pmatrix}, \forall R_m \in \mathcal{R}. \tag{29}$$

Its second-order mixed partial derivative is as follow:

$$\frac{\partial^2 Q}{\partial {F_{m,k}^{\text{rsu}}}^2} = \sum_{v_n\in\mathcal{V}} \frac{4c_k}{{F_{m,k}^{\text{rsu}}}^3} > 0. \tag{30}$$

For the objective function Q, it is known that $H(Q)$ is a positive definite matrix and the objective function Q is a convex function with respect to $F_{m,k}^{\text{rsu}}$. Consequently, in accordance with Benders partitioning theorem, the original MINLP can be decomposed into a two-stage problem [33]. First solve the discrete combinatorial part (28) for fixed resource proportions, then solve the resulting continuous convex optimization (31) for the given assignment, and iterate until convergence.

$$\begin{aligned}&\min_{\mathcal{X},\mathcal{P}} Q = \min_{\mathcal{X},\mathcal{P}} \sum_{v_n\in\mathcal{V}} \left(\beta_1 T_n^{\text{total}} + \beta_2\gamma E_n^{\text{total}}\right) + \beta_3 U,\\ &s.t. \text{C1,C2,C3,C4,C7.}\end{aligned} \tag{31}$$

IV. Algorithm design

As analyzed in Section III, by leveraging Benders partitioning, the original problem is decomposed into a discrete decision component and a continuous resource allocation subproblem. The continuous subproblem is convex with respect to the resource allocation variables, which guarantees a globally optimal solution for any given discrete configuration. Therefore, the main challenge lies in efficiently exploring the discrete decision space rather than solving a fully coupled nonconvex problem.

To address this, we propose a Multi-RSU Distributed Shared Offloading (MRDSO) scheme with a hybrid two-stage optimization framework, in which discrete decisions and continuous variables are optimized in a coordinated and iterative manner rather than through a one-shot decomposition. In the first stage, tasks are classified based on repetition weights, and the TLCS algorithm constructs a structure-aware offloading strategy and computing sequence by exploiting task similarity,

thereby guiding the search toward promising regions of the solution space and avoiding poor initial configurations. In the second stage, given the offloading decisions, GA is employed to optimize resource allocation under complex constraints [34]. In addition to standard crossover and mutation operations, an adaptive crossover probability is incorporated to dynamically balance exploration and exploitation, enhancing global search capability while reducing the risk of premature convergence. By integrating structure-aware initialization with adaptive stochastic search, the proposed MRDSO scheme effectively improves solution quality while maintaining computational efficiency, leading to reduced system latency, energy consumption, and load imbalance.

A. Two Layer Collaborative Sharing (TLCS) algorithm

As the task latency is determined by the most time consuming sub-task, we design a repetition weight that prioritizes stronger computing resources for similar sub-tasks with high repetition and high computing demand from vehicles.

Definition 1 (The repetition weight for sub-task): To minimize system latency, we prioritize computing resources for sub-tasks with higher initiation frequency and larger data volume. Hence, we define a repetition weight to measure sub-tasks priority and determine their computing sequences on each vehicle. The repetition weight for sub-task q_k is given as follow:

$$\zeta_k = \frac{b_k}{\sum\limits_{q_k \in \mathcal{K}} b_k} \times \frac{c_k}{\sum\limits_{q_k \in \mathcal{K}} c_k}, \tag{32}$$

where b_k denotes the number of times sub-task q_k is initiated in the system, and c_k represents the data size of sub-task q_k. We use the set $\zeta = \{\zeta_k\}$ to represent the repetition weights of all types of sub-tasks.

By jointly considering node computing resources, load status and this repetition weight, we achieve maximum resource utilization. Therefore, the proposed TLCS algorithm first divides all sub-tasks into high-repetition and low-repetition categories, and then performs two layer offloading. High-repetition sub-tasks are directly handled by the RSU, while low-repetition sub-tasks are processed locally, forming a complete offloading scheme as detailed in Algorithm 1.

(1) Task classification

In order to efficiently handle overlapping task offloading, we prioritize sub-tasks based on their request frequency and data volume. Sub-tasks with higher repetition weights ζ , which indicate more requests and larger data volumes, are prioritized. We introduce a dynamic threshold $\theta = \max\left(\left\lfloor \frac{\sum\limits_{b_k \in \mathcal{B}} b_k}{\|K\|} \right\rfloor, 2\right)$ to classify sub-tasks into high repetition and low repetition categories dynamically. Sub-tasks with request counts exceeding θ are classified as high repetition, while others are low repetition. In the task classification phase, we traverse the task sets of all vehicles, count the requests, sort by repetition weight, and classify using the dynamic threshold. The high repetition and low repetition sub-tasks sets are denoted as $Task_{\text{high}}$ and $Task_{\text{low}}$, respectively.

Algorithm 1 Two Layer Collaborative Sharing (TLCS) algorithm

Require: Vehicle information $\mathcal{V}$, RSU information $\mathcal{R}$, sub-task information $\mathcal{K}$, initial resource allocation values $\mathcal{F}_0$, number of vehicle V, number of RSUs M, number of similar sub-task types K

Ensure: $\mathcal{X} = (-1,-1)^{V\times K}$, $\mathcal{P} = (-1)^V$

Initialize $Q^{\text{rsu}}_{\min}$,$Q^{\text{rsu}}_{\text{total}}$,$t^{\text{loc}}_{\min}$,$e^{\text{loc}}_{\min}$,$t^{\text{shared}}_{\min}$,$e^{\text{shared}}_{\min}$;
Prioritize tasks based on repetition weight ζ and classify similar sub-tasks in high repetition $Task_{\text{high}}$ and low repetition $Task_{\text{low}}$ using dynamic threshold θ;
for each q_k in $\mathcal{K}$ **do**
  W_{q_k} = Obtain the set of vehicles that generate sub-task q_k;
end for
for each q_k in $Task_{\text{high}}$ **do**
  $Q^{\text{rsu}}_{\min} = +\infty$,$Q^{\text{rsu}}_{\text{total}} = 0$;
  for each RSU R_m in $\mathcal{R}$ **do**
    for each v_n in $\mathbf{W}_{q_k}$ **do**
      Obtain cost $Q^{\text{rsu}}_{\text{total}}$ using formulas (31);
      if $Q^{\text{rsu}}_{\text{total}} < Q^{\text{rsu}}_{\min}$ **then**
        $Q^{\text{rsu}}_{\min} = Q^{\text{rsu}}_{\text{total}}$;
      end if
    end for
  end for
  $x^{\text{rsu}}_{v_{best},q_k,m} = 1$,$\mathcal{X}[v_{best}][q_k] = (1,m)$;
  $\mathbf{W}_{q_k} = \mathbf{W}_{q_k} - v_{best}$;
end for
for each q_k in $Task_{\text{low}}$ **do**
  $t^{\text{loc}}_{\min} = +\infty, e^{\text{loc}}_{\min} = +\infty$,$v_{best} = -1$;
  for each v_n in $\mathbf{W}_{q_k}$ **do**
    if $\beta_1 EFT_{v_n,q_k} + \beta_2 e^{\text{loc}}_{v_n,q_k} < \beta_1 t^{\text{loc}}_{\min} + \beta_2 e^{\text{loc}}_{\min}$ **then**
      $v_{\text{best}} = v_n$;
    end if
  end for
  $P[v_{\text{best}}] \leftarrow q_k$;
  $x^{\text{loc}}_{v_{\text{best}},q_k} = 1$,$\mathcal{X}[v_{\text{best}}][q_k] = (0,n)$;
  $\mathbf{W}_{q_k} = \mathbf{W}_{q_k} - v_{best}$;
end for
for each q_k in $\mathcal{K}$ **do**
  for each v_n in $\mathbf{W}_{q_k}$ **do**
    $t^{\text{shared}}_{\min} = +\infty, e^{\text{shared}}_{\min} = +\infty$,$v_{best} = -1$;
    for each v_j in $\mathcal{V}$ and $(\mathcal{X}[v_j][q_k][1] \neq -1$ *or* $\mathcal{X}[v_n][q_k][1] \neq 2)$ **do**
      Obtain latency and energy consumption via V2V or R2V sharing using formulas in shared offloading;
      if $\beta_1 t^{\text{shared}}_{v_n,q_k,v_j} + \beta_2 e^{\text{shared}}_{v_n,q_k,v_j} < \beta_1 t^{\text{shared}}_{\min} + \beta_2 e^{\text{shared}}_{\min}$ **then**
        $v_{\text{best}} = v_j$;
      end if
    end for
    $\mathcal{X}[v_j][q_k] = (0, v_{best})$;
  end for
end for
return $\mathcal{X}$, $\mathcal{P}$

(2) Offloading strategy for high repetition sub-tasks
For sub-task q_k in $Task_{\text{high}}$, we use the preset resource allocation cycle F_0 as the baseline, and greedily choose R_{best} based on the weighted cost of latency, energy consumption, and the degree of load on the compute unit. That minimizes the cost for these sub-tasks. If more than one vehicle in the R_{best} coverage area contains the sub-task, we will determine the vehicle with the shortest transmission distance as the task uploading vehicle to transmit the task data to R_{best}, denoted as $v_{\text{best}}^{\text{tran}}$. If other TVs are not in the communication range of R_{best}, the results are first transmitted over a wired link between RSUs and then shared with these vehicles via R2V communication.

(3) Offloading strategy for low repetition sub-tasks
For sub-task q_k in $Task_{\text{low}}$, considering their lower initiation frequency, we opt for local computing to reduce RSU load. First, we traverse vehicles generating these sub-tasks and greedily select the vehicle v_{best} that minimizes the cost of $Task_{\text{low}}$. If multiple sub-tasks are computing on the same vehicle, their computing sequences is determined by comparing their repetition weights. Finally, we check if other vehicles generating this sub-tasks are in the communication range of v_{best}. If not, v_{best} transmits results to its associated $R_{m'}$ via V2R, then to other RSUs via fiber-optic links, and finally to the target vehicles via R2V. If they are in range, results are shared directly via V2V communication.

B. Resource Allocation based on the Genetic Algorithm (GA)

Using the TLCS algorithm from the previous section, we obtained the offloading strategy $\mathcal{X}$ and computing sequence $\mathcal{P}$ based on the initial resource allocation frequency $\mathcal{F}_0$. In this section, we use resource allocation based on the Genetic Algorithm (GA) to further optimize RSU resource allocation based on $\mathcal{X}$, aiming to find the optimal resource allocation strategy $\mathcal{F}$. The optimization subproblem is as problem (28).

To solve the above problem using GA, each chromosome represents a resource allocation strategy, encoded as a floating-point number for proportion. Algorithm 2 outlines the detailed algorithmic flow and its stages as follows.

(1) Generate the initial population
Generate an initial population $\mathcal{P}^{(0)}$ of size P_{size}, where each individual is a resource allocation matrix F whose gene length equals the total number K of sub-task types across all RSUs. Provide a legal initial $\mathcal{F}_0$ to each RSU based on the resource scaling constraint.

(2) Set the fitness function
For every matrix F in the current population $\mathcal{P}^{(g-1)}$, compute its fitness using formula (33). The fitness function comprises two parts, *i.e.* the objective function from this paper and a penalty function. The penalty function ensures two constraints. First, each sub-task gets computing resources. Second, the total allocated resources don't exceed the RSU's maximum capacity.

$$fitness\,(F) = Q + \sum_{R_m \in \mathcal{R}} \left[\lambda_{\text{sum}} \cdot \left(\sum_{q_k \in \mathcal{K}} F_{m,k}^{\text{rsu}} - F_m^{\text{rsu}} \right)^2 + \sum_{q_k \in \mathcal{K}} \lambda_{\min} \cdot e^{F_m^{\min} - F_{m,k}^{\text{rsu}}} \right], \tag{33}$$

where λ_{sum} and $\lambda_{\min}$ are the penalty coefficients for sum and minimum resource constraints, respectively. $F_m^{\min}$ denotes the minimum computing resources allocated to each sub-tasks type by RSU R_m.

(3) Selection
We use an elite selection strategy, carry over the top 20% individuals from the current population $\mathcal{P}^{(g-1)}$ into the next generation $\mathcal{P}^{(g)}$, denoted as Pelite. The remaining individuals will undergo crossover and mutation.

(4) Crossover and mutation
Crossover involves recombining resource allocation matrices from two parent individuals to produce offspring, while mutation modifies an individual's genes at a probability p_{m}, adjusting resource allocation for a single RSU. To prevent the algorithm from getting stuck in local optima, we dynamically adjust the crossover probability based on iteration count. Starting with a high value to encourage global exploration and gradually decreasing it to focus on exploitation as iterations proceed. The dynamic crossover probability is calculated using the formula as follow:

$$p_{\text{c}} = p_{\text{c,max}} - (p_{\text{c,max}} - p_{\text{c,min}}) \cdot \frac{\omega}{\omega_{\max}}, \tag{34}$$

where ω denotes the current iteration, $\omega_{\max}$ is the total iterations, $p_{\text{c,max}}$ the initial maximum crossover probability, and $p_{\text{c,min}}$ the final minimum crossover probability.

(5) Termination condition judgment
If the fitness value change is below a set threshold or the iteration count hits N_{iter}, stop the process. Decode the fittest chromosome in the population to get the optimal resource allocation proportions.

C. Complexity Analysis

We analyze the time complexity of the two algorithms. Algorithm 1 comprises four main components. The time complexity of task classification and sorting is $O\,(K \cdot \log K)$, the time complexity of high repetition similar sub-task is $O\,(M \cdot K \cdot V)$, the time complexity of low repetition similar sub-task is $O\,(K \cdot V)$, the time complexity of non-optimal similar sub-task is $O\,(M \cdot K \cdot V)$, where M is the number of RSUs, K is the number of similar sub-task types, and V is the number of vehicles. Hence, the worst-case complexity of Algorithm 1 is $O\,(M \cdot K \cdot V)$, though in practice it seldom reaches this bound. Algorithm 2 is dominated by the genetic algorithm. A single fitness evaluation of the resource allocation matrix costs $O\,(M \cdot K \cdot P_{\text{size}})$, selection costs $O\,(P_{\text{size}})$, and crossover plus mutation cost $O\,(M \cdot K \cdot P_{\text{size}})$, with P_{size} denoting the population of allocation matrices. After N_{iter} iterations, the

Algorithm 2 Resource Allocation based on the Genetic Algorithm (GA)

Require: RSU set $\mathcal{R}$, offloading strategy X, crossover probability P_c, retention probability P_r, mutation probability P_m, max iterations N_{iter}, population size P_{size}, number of RSUs M, number of similar sub-task types K

Ensure: optimal resource allocation matrix $\mathbf{F}^*$

initialize population $\mathcal{P}^{(0)} \leftarrow \{\mathbf{F}_1, \ldots, \mathbf{F}_{P_{\text{size}}}\}$ with $\mathbf{F}_i \in [0,1]^{M\times K}$ uniformly at random;
$\mathbf{F}^* \leftarrow \arg\max_{\mathbf{F}\in\mathcal{P}^{(0)}} \text{fitness}(\mathbf{F})$;
for generation $g = 1$ **to** N_{iter} **do**
  evaluate fitness for every $\mathbf{F} \in \mathcal{P}^{(g-1)}$ via Eq. (33);
  select top-$\lfloor P_r P_{\text{size}} \rfloor$ individuals into $\mathcal{P}_{\text{elite}}$;
  $\mathcal{P}^{(g)} \leftarrow \mathcal{P}_{\text{elite}}$;
  while $|\mathcal{P}^{(g)}| < P_{\text{size}}$ **do**
    randomly pick parents $\mathbf{F}_a, \mathbf{F}_b$ from $\mathcal{P}^{(g-1)}$;
    if rand() $< P_c$ **then**
      create offspring $\mathbf{F}'$ by uniform crossover on $(\mathbf{F}_a, \mathbf{F}_b)$;
    else
      $\mathbf{F}' \leftarrow \mathbf{F}_a$;
    end if
    if rand() $< P_m$ **then**
      mutate entries of $\mathbf{F}'$ randomly within $[0,1]$;
    end if
    repair column-sum of $\mathbf{F}'$ to 1 if needed;
    $\mathcal{P}^{(g)} \leftarrow \mathcal{P}^{(g)} \cup \{\mathbf{F}'\}$;
  end while
  update $\mathbf{F}^* \leftarrow \arg\max_{\mathbf{F}\in\mathcal{P}^{(g)}} \text{fitness}(\mathbf{F})$;
end for
return $\mathbf{F}^*$

overall complexity of Algorithm 2 is $O\left(M \cdot K \cdot P_{\text{size}} \cdot N_{\text{iter}}\right)$. Finally, since Algorithm 1 is absorbed into Algorithm 2, the entire MRDSO scheme is governed by the genetic main loop, yielding a complexity of $O\left(M \cdot K \cdot P_{\text{size}} \cdot N_{\text{iter}}\right)$, which remains low.

V. Results and discussion

A. Parameter Settings

This paper considers a representative traffic scenario in typical road environments to address RSU load imbalance. Vehicle-generated tasks are randomly distributed to ensure the generality of the results. The main parameter settings in the TABLE III are determined based on widely adopted configurations in the VEC literature and the characteristics of the considered scenario. General system parameters, including communication bandwidth, transmission power, computing capacities of vehicles and RSUs, and related system configurations, are selected according to commonly used values in existing studies to ensure practical relevance and comparability [35]–[37].

TABLE III
Main parameters throughout simulation.

Parameter	Definition	Value
N	Number of vehicles	$[20, 30]$
M	Number of RSUs	$[3, 6]$
K	Number of sub-tasks types	$[20, 30]$
d_k	Data size of sub-task q_k	$[1000, 2000]$kbits
c_k	Computing resources required for sub-task q_k	$[3, 9] \times 10^8$cycle
f_n	Computing capacity of vehicle v_n	$[0.8, 1.2] \times 10^9$cycle/s
$f_{\max}^{\text{mec}}$	Computing capacity of RSU	$[6, 16] \times 10^9$cycle/s
p_n	Transmission power	0.1W
ρ_0	Gaussian noise density	-114dBm
p_1	Fading factor of the link channel	3
B^{V2R}	Bandwidth of V2R communication	100Mhz
B^{V2V}	Bandwidth of V2V communication	20Mhz
$\|\|\mathbf{M}_n\|\|$	Sub-tasks number generated by vehicle v_n	$[3, 5]$
R_{rsu}	Communication range of the RSU	500m
R_{veh}	Communication range of vehicle v_n	200m
β_1	Weighting coefficient for latency	0.6
β_2	Weighting coefficient for energy consumption	0.2
β_3	Weighting coefficient for load balancing	0.2

B. Simulation Results and Analysis

To verify the effectiveness of the proposed MRDSO scheme, we compare it with four benchmark schemes under a unified system model and consistent optimization settings to ensure a fair evaluation, without introducing additional learning-based components. By varying key parameters such as RSU computing resources, task data size, and sub-task types, we evaluate the system-average cost to demonstrate the performance of the MRDSO scheme.

(1) Equal Allocation Offloading (EAO): use the TLCS algorithm for multi-RSU shared offloading and allocate resources equally.
(2) Selective Shared Offloading (SSO): shared offloading is considered only within the communication range of the task vehicle compared with our proposed MRDSO scheme [15].
(3) Load-Aware Collaborative Offloading (LACO): ignores shared offloading and instead considers multi-RSU collaboration to balance latency, energy, and degree of load [25].
(4) Random Offloading (RO): randomly offload tasks to RSUs, and allocate resources to RSUs using GA.

Fig. 2 demonstrates the impact of changes in RSU computing capacity on the average cost of each scheme. As the RSU computing capacity is increased, the cost of each scheme shows a decreasing trend, but the cost curve flattens out when the RSU computing resource reaches 1.2×10^{10} cycles/s, indicating that there is a diminishing marginal benefit from purely increasing the RSU computing capacity after it reaches a certain threshold, and that more is not necessarily better. At this point, the proposed MRDSO scheme reduces

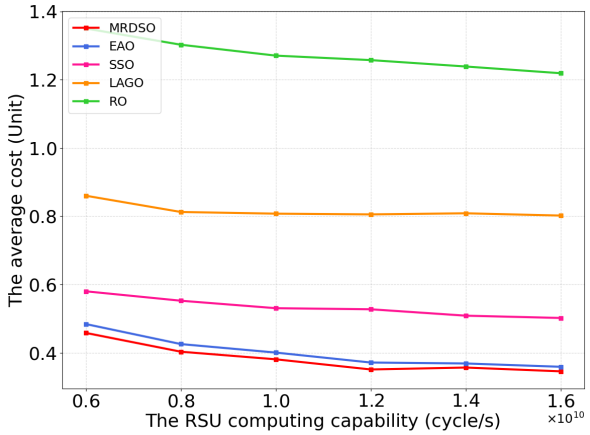

Fig. 2. Relationship between cost and RSU computing capacity.

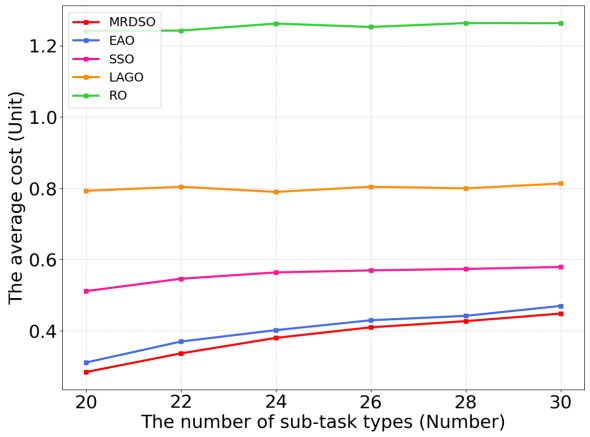

Fig. 3. Relationship between cost and the number of sub-tasks.

the system cost by approximately 5.5%, 33.4%, 56.4%, and 72.1% compared to EAO, SSO, LAGO, and RO, respectively. Overall, the MRDSO scheme achieves the lowest average cost through shared offloading and in particular delivers the greatest performance gain over the RO scheme. Even as RSU computing resource continues to grow and the cost curves of all schemes level off, MRDSO by eliminating redundant computations of overlapping tasks with multi-RSU collaboration maintains higher computing efficiency and lower node energy consumption, keeping its cost lower than SSO, LACO and RO schemes.

Fig. 3 shows the effect of changes in the number of sub-task types on the average cost when the total number of sub-tasks is fixed. Overall, thanks to the shared offloading of similar sub-tasks, the average costs of the MRDSO, EAO and SSO schemes all decrease as the number of sub-task types is reduced with the repetition rate increases, whereas the RO and LACO schemes, which do not eliminate redundant computing for similar sub-tasks, exhibit no clear trend with the type number. When the number of sub-task types is 30, the proposed MRDSO scheme reduces the system cost by approximately 4.6%, 22.6%, 44.9%, and 64.5% compared to EAO, SSO, LAGO, and RO, respectively. Benefiting from

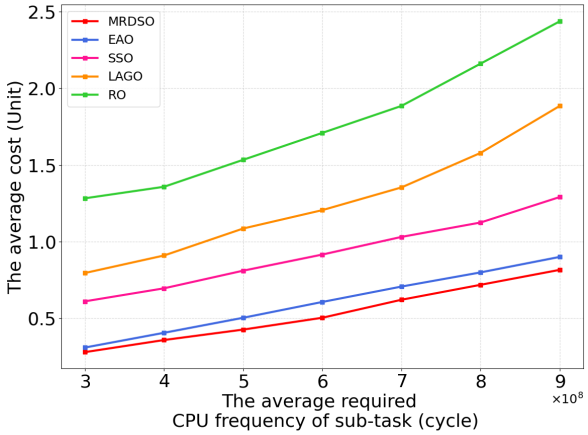

Fig. 4. Relationship between cost and the average required CPU cycle of sub-tasks.

system-wide shared offloading, MRDSO schemes and EAO schemes reduce redundant computing among similar sub-tasks faster and more comprehensively than SSO scheme. Consequently, their costs respond more promptly and drop more sharply to changes in the number of sub-task types.

Fig. 4 shows how the average cost changes with the average required CPU cycles of sub-tasks when all other conditions remain constant. In general, the average cost of all the scenarios shows an increasing trend as the CPU cycles required by the sub-tasks increase. This is because with the same total amount of resources owned by the system, as the CPU cycles required by the sub-tasks increase, it increases the computing burden of the system, which leads to an increase in the computing latency and energy consumption of the tasks in the system in order to affect the performance of the individual scenarios. The MRDSO scheme has a lower average cost compared to other schemes. When the average required CPU frequency of sub-task is 9×10^8 cycle, the proposed MRDSO scheme reduces the system cost by approximately 9.4%, 36.8%, 56.7%, and 66.5% compared to EAO, SSO, LAGO, and RO, respectively. In particular, the increase in CPU cycles required for sub-tasks saves more latency and energy consumption for shared offloading compared to RO, LACO, and SSO schemes, and the MRDSO scheme will show more and more advantages in terms of average cost.

Fig. 5 shows how the average cost varies with the number of RSUs when the total average CPU cycles are kept constant. Overall, for every scheme, increasing the number of RSUs first lightens each RSU's load and then expands the system's computing resources, reducing sub-task offloading latency and driving down the average cost.The MRDSO scheme has a lower average cost than the other scenarios. When the number of RSU is 6, the proposed MRDSO scheme reduces the system cost by approximately 4.6%, 44.9%, 46.1%, and 63.3% compared to EAO, SSO, LAGO, and RO, respectively. Compared with SSO, LACO and RO schemes, MRDSO scheme starts from a near optimal offloading state, so it is the least sensitive to cost reductions brought by adding more RSUs, thanks to its early elimination of redundant computation for overlapping

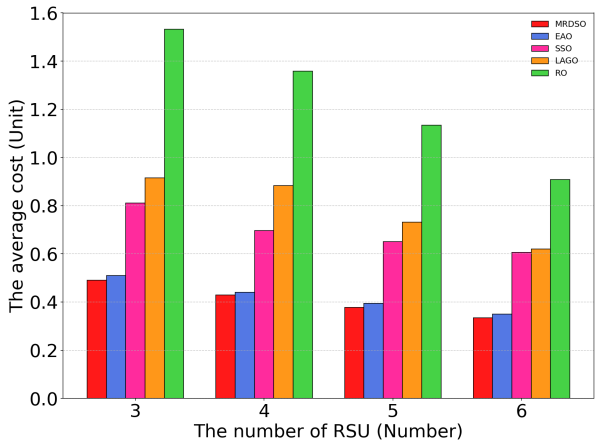

Fig. 5. Relationship between cost and the number of RSU.

tasks through multi RSU collaboration.

VI. Conclusion

In this paper, we solve the task offloading and resource allocation problem of overlapping tasks in a multi-user, multi-RSU scenario. Our objective is to minimize system average latency and energy consumption while balancing the load across RSUs. We propose a multi-RSU distributed offloading strategy. It uses TLCS algorithm for offloading strategy and computing sequences on vehicles, and GA for optimizing resource allocation among RSUs, thus reducing system costs. Extensive simulations show the impact of various factors on costs and confirm the effectiveness of our scheme. Compared to others, it significantly reduces system costs and balances RSU loads. However, this work focuses on sub-tasks that are independent and can be executed in parallel. In future work, task dependencies will be incorporated into the proposed framework by introducing precedence constraints and corresponding scheduling mechanisms, thereby enhancing its capability to model more complex application scenarios.

References

[1] Z. Yu, Y. Zhao, T. Deng, L. You, and D. Yuan, "Less carbon footprint in edge computing by joint task offloading and energy sharing," *IEEE Networking Letters*, vol. 5, no. 4, pp. 245–249, 2023.

[2] D. Cao, K. Zeng, J. Wang, P. K. Sharma, X. Ma, Y. Liu, and S. Zhou, "Bert-based deep spatial-temporal network for taxi demand prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9442–9454, 2022.

[3] B. Lin, Q. Chen, X. Chen, W.-K. Jia, Y. Lu, and N. N. Xiong, "Dqps: An intelligent multi-hop computation offloading scheme for workflow applications in vehicular edge computing networks," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 1, pp. 2202–2216, 2025.

[4] Y. Sun, Z. Wu, K. Meng, and Y. Zheng, "Vehicular task offloading and job scheduling method based on cloud-edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 14 651–14 662, 2023.

[5] D. Cao, M. Wu, N. Gu, R. S. Sherratt, U. Ghosh, and P. K. Sharma, "Joint optimization of computation offloading and resource allocation considering task prioritization in isac-assisted vehicular network," *IEEE Internet of Things Journal*, vol. 11, no. 18, pp. 29 523–29 532, 2024.

[6] C. Tang, G. Yan, H. Wu, and C. Zhu, "Computation offloading and resource allocation in failure-aware vehicular edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1877–1888, 2024.

[7] R. W. Coutinho and A. Boukerche, "Design of edge computing for 5g-enabled tactile internet-based industrial applications," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 60–66, 2022.

[8] X. Dai, Z. Xiao, H. Jiang, M. Lei, G. Min, J. Liu, and S. Dustdar, "Offloading dependent tasks in edge computing with unknown system-side information," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 4345–4359, 2023.

[9] H. Qian, X. Shi, A. Hawbani, Y. Bi, Z. Liu, A. Muthanna, and L. Zhao, "Tracer: Transfer knowledge-based collaborative vehicle trajectory prediction for highway traffic toward cross-region adaptivity," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 7, pp. 10 747–10 763, 2025.

[10] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[11] Z. Ning, K. Zhang, X. Wang, and L. Guo, "Cooperative service caching and peer offloading in internet of vehicles based on multi-agent meta-reinforcement learning," *Tongxin Xuebao/Journal on Communications*, vol. 42, pp. 118–130, 06 2021.

[12] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2d-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for mec," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 4858–4873, 2021.

[13] C. Xu, J. Guo, Y. Li, H. Zou, W. Jia, and T. Wang, "Dynamic parallel multi-server selection and allocation in collaborative edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 523–10 537, 2024.

[14] L. Zhao, L. Sun, A. Hawbani, Z. Liu, X. Tang, and L. Xu, "Dual dependency-aware collaborative service caching and task offloading in vehicular edge computing," *IEEE Transactions on Mobile Computing*, vol. 24, no. 10, pp. 10 963–10 977, 2025.

[15] Z. Liao, Z. Shao, B. Zheng, and X. Tang, "De-duplicated hierarchical offloading in vehicular edge computing with task dependencies," *IEEE Internet of Things Journal*, vol. 12, no. 8, pp. 10 293–10 303, 2025.

[16] L. Zhao, Z. Zhao, E. Zhang, A. Hawbani, A. Y. Al-Dubai, Z. Tan, and A. Hussain, "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3386–3400, 2023.

[17] H. Zhou, M. Li, N. Wang, G. Min, and J. Wu, "Accelerating deep learning inference via model parallelism and partial computation offloading," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 475–488, 2023.

[18] Y. Liang, H. Tang, H. Wu, Y. Wang, and P. Jiao, "Lyapunov-guided offloading optimization based on soft actor-critic for isac-aided internet of vehicles," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 708–14 721, 2024.

[19] H. Lin, B. Xiao, X. Zhou, Y. Zhang, and X. Liu, "A multi-tier offloading optimization strategy for consumer electronics in vehicular edge computing," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 1, pp. 2118–2130, 2025.

[20] S. Lei, H. Tang, C. Li, X. Zhang, C. Xu, and H. Wu, "Federated maddpg-based collaborative scheduling strategy in vehicular edge computing," *IEEE Transactions on Mobile Computing*, vol. 25, no. 1, pp. 54–66, 2026.

[21] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in iot," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763–9773, 2021.

[22] W. Zhao, Y. Cheng, Z. Liu, X. Wu, and N. Kato, "Asynchronous drl-based multi-hop task offloading in rsu-assisted iov networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 1, pp. 546–555, 2025.

[23] S. Li, N. Zhang, H. Chen, S. Lin, O. A. Dobre, and H. Wang, "Joint road side units selection and resource allocation in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 190–13 204, 2021.

[24] Y. Cui, H. Li, D. Zhang, A. Zhu, Y. Li, and H. Qiang, "Multiagent reinforcement learning-based cooperative multitype task offloading strategy for internet of vehicles in b5g/6g network," *IEEE Internet of Things Journal*, vol. 10, no. 14, pp. 12 248–12 260, 2023.

[25] Z. Zhang and F. Zeng, "Efficient task allocation for computation offloading in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5595–5606, 2023.

[26] Y. Gao, H. Zhang, F. Yu, Y. Xia, and Y. Shi, "Joint computation offloading and resource allocation for mobile-edge computing assisted ultra-dense networks," *Journal of Communications and Information Networks*, vol. 7, no. 1, pp. 96–106, 2022.

[27] X. Deng, J. Yin, P. Guan, N. N. Xiong, L. Zhang, and S. Mumtaz, "Intelligent delay-aware partial computing task offloading for multiuser industrial internet of things through edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 2954–2966, 2023.
[28] C. Liu and K. Liu, "Toward reliable dnn-based task partitioning and offloading in vehicular edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3349–3360, 2024.
[29] D. Cao, N. Gu, M. Wu, and J. Wang, "Cost-effective task partial offloading and resource allocation for multi-vehicle and multi-mec on b5g/6g edge networks," *Ad Hoc Networks*, vol. 156, p. 103438, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870524000490
[30] Q. Chen, Z. Kuang, and L. Zhao, "Multiuser computation offloading and resource allocation for cloud–edge heterogeneous network," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3799–3811, 2022.
[31] Y. Fu, X. Wang, and F. Fang, "Multi-objective multi-dimensional resource allocation for categorized qos provisioning in beyond 5g and 6g radio access networks," *IEEE Transactions on Communications*, vol. 72, no. 3, pp. 1790–1803, 2024.
[32] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot generalized task assignment problem," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4765–4771.
[33] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.
[34] X. Wang, M. Yi, M. Chen, Z. Liu, A. V. Vasilakos, H. Herbert Song, and A. Farouk, "Joint task offloading and resource allocation in ris-assisted noma-vec intent-based networking," *IEEE Internet of Things Journal*, pp. 1–1, 2025.
[35] D. Cao, Y. Yang, Y. Wang, P. K. Sharma, O. Alfarraj, A. Tolba, and M. Zhu, "A reservation-based multi-source distributed offloading strategy in dynamic environments," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1866–1876, 2024.
[36] J. Wu, Y. Zou, X. Zhang, J. Liu, W. Sun, and G. Du, "Dependency-aware task offloading strategy via heterogeneous graph neural network and deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 12, no. 13, pp. 22 915–22 933, 2025.
[37] K. Zhang, X. Gui, D. Ren, and D. Li, "Energy–latency tradeoff for computation offloading in uav-assisted multiaccess edge computing system," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6709–6719, 2021.

Dun Cao (Member, IEEE) received the B.S. degree from Central South University in 2001, the M.S. degree from Hunan University in 2006, and the Ph.D. degree from the Changsha University of Science and Technology in 2017. She is currently a Professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology. She has published more than 40 journal and conference papers. Her research interests include vehicular networks and MIMO wireless communications.

Chi Peng is currently pursuing the M.S. degree in the Changsha University of Science and Technology. His current research interests is mobile edge computing for Internet of Vehicles.

Bo Peng is currently pursuing the M.S. degree in the Changsha University of Science and Technology. His current research interests is mobile edge computing for Internet of Vehicles.

Penglu Liu Penglu Liu received the B.S. degree in electronic information engineering from the SIAS International College, Zhengzhou University, Zhengzhou, China, in 2014, the M.S. degree in signal and information processing from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2017, and the Ph.D. degree in information and communication engineering from the Northwestern Polytechnical University, Xi'an, China, in 2022. From 2021 to 2022, he was a visiting Ph.D. student with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, Canada. From 2023 to 2025, he was a lecturer with the School of Computer and Communication Engineering, Changsha University of Science and Technology (CSUST), Changsha, China. He is currently a lecturer with the School of Physics and Electronic Science, CSUST. His research interests include multi-antenna wireless communication, multiple access technologies, near-field communication, mobile edge computing, convex optimization theory, and deep reinforcement learning.

R. Simon Sherratt (M97-SM02) was born in Heswall, UK. He received his B.Eng. degree in Electronic Systems and Control Engineering from Sheffield City Polytechnic UK in 1992, M.Sc. degree in data telecommunications and Ph.D. degree in video signal processing from the University of Salford in 1994 and 1996. Since 1996, he has been a Lecturer in Electronic Engineering at the University of Reading where he is now a senior lecturer in consumer electronics and currently head of Electronic Engineering. His research topic is signal processing in consumer electronic devices concentrating on equal- ization, communications layer 1, DSP architectures and adaptive signal processing. Eur Ing Dr. Sherratt is a senior member of the IEEE, member of the IEEE Consumer Electronics Society AdCom (2003-2005, 2006-2008) holding the International Symposium on Consumer Electronics Liaison officer post (2005-) and the Society awards chair post (2006-), member of the IEEE Transactions on Consumer Electronics publications committee (2004-), IEEE International Conference on Consumer Electronics ExCom and vice-technical chair (2006), chair of the IEEE International Symposium on Consumer Electronics 2004 and committee member for 2002, 2003, 2005 and 2006, founder and current chair of the IEEE UKRI Consumer Electronics and Broadcast Technology Joint Chapter. He received the IEEE Chester Sall 1st place best Transactions paper award in 2004 and the best paper in the IEEE International Symposium on Consumer Electronics 2006.

Jin Wang (Senior Member, IEEE) received the B.S. and M.S. degree from Nanjing University of Posts and Telecommunications, China in 2002, 2005 respectively. He received Ph.D. degree from Kyung Hee University Korea in 2010. Now, he is a professor at Hunan University of Science and Technology. He has published more than 300 international journal and conference papers. His research interests mainly include wireless sensor network, network performance analysis and optimization. He is a Fellow of IET, and senior member of IEEE.