Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2954987, 13 pages
https://doi.org/10.1155/2022/2954987

Research Article

Joint Optimization of Offloading and Resource Allocation for SDN-Enabled IoV

Li Lin and Lei Zhang

College of Information Science and Technology, Donghua University, Shanghai 201620, China

Correspondence should be addressed to Lei Zhang; lei.zhang@dhu.edu.cn

Received 24 December 2021; Revised 12 February 2022; Accepted 21 February 2022; Published 4 March 2022

Academic Editor: Changqing Luo

With the development of various vehicle applications, such as vehicle social networking, pattern recognition, and augmented reality, diverse and complex tasks have to be executed by the vehicle terminals. To extend the computing capability, the nearest roadside-unit (RSU) is used to offload the tasks. Nevertheless, for intensive tasks, excessive load not only leads to poor communication links but also results to ultrahigh latency and computational delay. To overcome these problems, this paper proposes a joint optimization approach on offloading and resource allocation for Internet of Vehicles (IoV). Specifically, assuming particle tasks assigned for vehicles in the system model are offloaded to RSUs and executed in parallel. Moreover, the software-defined networking (SDN) assisted routing and controlling protocol is introduced to divide the IoV system into two independent layers: data transmission layer and control layer. A joint approach optimized offloading decision, offloading ratio, and resource allocation (ODRR) is proposed to minimize system average delay, on the premise of satisfying the demand of the quality of service (QoS). By comparing with conventional offloading strategies, the proposed approach is proved to be optimal and effective for SDN-enabled IoV.

1. Introduction

The Internet of Vehicles (IoV) is one of the most promising application for Internet of Things (IoT) technology. Currently, it is fuelled by advances in related technology and industries, such as the decline in sensor costs, the widespread wireless connections, the improvement of computing capabilities, the development of cloud computing, and wireless transmission and positioning. In IoV, large amounts of heterogeneous data (such as voice, program code, and video) need to be transmitted in various links, e.g., vehicles to vehicles, vehicles to roadside equipment, and vehicles to the cloud [1–4]. These requirements are challenging for cloud infrastructure and wireless access networks. Upgraded service such as ultralow latency, continuity of user experience, and high reliability have been proposed to highly promote the local services at the edge of the network close to the terminals. The basic idea of Mobile Edge Computing (MEC) is to migrate the cloud-computing platform to the edge of the mobile access network. The traditional cellular network is deeply integrated with Internet services, to reduce the end-to-end delay of mobile service delivery and improve the user experience. With the development of MEC, mobile devices with limited resources can implement various new applications by offloading computing tasks to the MEC server, such as autonomous driving, augmented reality, and image processing [5–7]. The MEC server is owned by the network operator and is directly implemented in cellular base stations (BSs) or local wireless access points (APs) as a general-purpose computing platform [8, 9]. On the one hand, higher requirements on computing resources and storage capacity and capabilities consumption are put forward to carry the various vehicular applications. On the other hand, the computing power of in-vehicle devices is limited by the size and portability. Therefore, MEC with distributed computing capabilities, rich computing resources, and flexible wireless accessibility is promising for IoV. For instance, the selected part of the in-vehicle task can be

offloaded to the appropriate MEC for parallel processing and feedback the execution result to the vehicle-mounted terminal through the neighboring BSs or APs.

Although the MEC server has richer resources than local equipment, excessive load has a great impact on the transmission link. Considering the transmission delay on the link [10], this is detrimental to delay-sensitive tasks. As the number of vehicles increases, and the communication environment becomes worse, resulting in high transmission delays. In addition, the heterogeneous nature of vehicle-mounted tasks places higher requirements on the entire system. There are two types of separable task offloading: (1) One is a bit-type task, which can be arbitrarily divided into several independent parts [11, 12], and these parts can be processed in parallel on different platforms. (2) The other is a code-oriented task, which is composed of various components [13, 14]. There are dependencies between task components and need to be executed in an orderly or continuous manner. Therefore, a reliable computing offloading solution is needed to support low-latency, highly reliable IoV services.

The edge computing capabilities of nearby RSUs have been leveraged to meet the task-intensive requests and strict latency requirements, i.e., part or all of the divisible bit-type tasks with high delay requirements are selected to be offloaded to nearby RSUs; then, the computing delay can be greatly reduced by parallel computing. Nevertheless, the unified task intensity is not often, some RSUs may have to handle extra requests beyond their capabilities, while other RSUs are relatively idle. Benefiting from the software-defined network (SDN) architecture, SDN controllers with global information are able to coordinate edge computing resources [15, 16]. Combined with the current global situation, the requested tasks are controlled and forwarded to the corresponding target nodes through the SDN controller, which can effectively integrate global resources.

A lot of research work has focused on task offloading in IoV [17–20], majorly considering one part of offloading strategy, offloading ratio or resource allocation, and lack of the utilization of SDN to efficiently solve the load balancing problem. In this paper, we jointly optimize the offloading strategy, offloading ratio, and resource allocation, in order to minimize the system delay of SDN enabled IoV. Moreover, the effects of different task complexity on transmission and execution are also considered. The main contributions of this paper are as follows:

(i) A tasks-divisible system model with a software-defined network is proposed based on two-layer transmission

(ii) A Particle Swarm Optimization- (PSO-) based heuristic approach for the overall optimization is proposed, which can effectively solve the offloading strategy problem of multiuser and multiobjective nodes. This approach works by decomposing the problem into three subproblems: (1) offloading decision of vehicles; (2) resource allocation by RSUs; and (3) offload ratio of vehicles. It greatly reduces the complexity of the problem and is very effective for solving the multivehicle and multimec offloading scenario

(iii) The offloading decisions, local offloading ratios, and resource allocation (ODRR) are jointly optimized in a complete way, to maximize the system performance

By comparing with the conventional offloading strategies, simulation results show the proposed ODRR approach achieves the best performance.

2. Related Work

The offloading strategy has been a hot research topic for IoV, and various offloading models have been proposed in different application scenarios. Considering the standby time of mobile devices and the latency sensitivity of tasks, most work in edge computing or fog computing focuses on the optimization of energy consumption or latency. Therefore, we survey the related work according to the optimization goals, as shown in Table 1.

2.1. Optimizing the Energy Consumption. For mobile users, processing various application tasks consumes a lot of energy, so improving the standby time of the device has always been a concern. Some work is devoted to reducing local computing power consumption to improve standby time. The energy cost of task calculation and file transmission has been studied in [21]. Combining the multiaccess characteristics of 5G heterogeneous networks, jointly optimizing offloading and wireless resource allocation to minimum energy consumption within delay constraints. A multiuser MEC system with wireless power supply has been modeled in [22], in order to solve a practical scenario that requires delay limit and reduces the total energy consumption of the AP, jointly optimizing the energy transmission beamforming of the access point, the frequency of the central processing unit, the number of bits offloaded by users, and the time allocation among users. [23] has proposed an energy-optimal dynamic computation offloading scheme algorithm to minimize system energy consumption under energy overhead and other resource constraints.

2.2. Optimizing the Delay of the System. For latency-sensitive tasks, researchers are devoted to reducing system latency and improving user experience by optimizing local computing resources and edge node resources. [10] aim to minimize the maximum delay among users by joint optimization of offloading decision, computing resource allocation and resource block and power. [17] investigate the calculation rate maximization problem in the MEC wireless power supply system enabled by UAV, which is subject to the energy harvesting causal constraints and the UAV speed constraints. A new device-to-device multiassistant MEC system that requests local users to nearby auxiliary devices for collaborative computing has been designed in [18]. By optimizing task allocation and resource allocation to minimize the execution delay, a collaborative method has been

Table 1: Summary and comparison of the most relevant references.

Work	MEC/fog	User	Load balancing	Partial offloading	Resource allocation by computing nodes	Optimizing goals
Du et al. [10]	Multiple	Multiple	Yes	—	Yes	Minimize the total energy consumption of the system with delay constraints.
Zhang et al. [21]	Multiple	Multiple	—	—	—	Minimize the total energy consumption of the system with delay constraints.
Wang et al. [22]	Single	Multiple	—	—	Yes	Minimize the total energy consumption
Zhou et al. [17]	Single	Multiple	—	—	Yes	Maximize the calculation rate.
Xing et al. [18]	Multiple	Single	—	—	—	Minimize the total delay of the system with energy constraints.
Tran and Pompili [19]	Multiple	Multiple	—	—	Yes	Minimize the total energy consumption and the delay of the system.
Zhou et al. [20]	Single	Multiple	—	Yes	Yes	Minimize the maximal task completion time.
Chen et al. [23]	Multiple	Multiple	—	Yes	—	Minimize the total energy consumption of the system with delay and energy consumption constraints.
Chen et al. [24]	Multiple	Multiple	Yes	Yes	—	Minimize the competition latency of the task with maximum delay
Wang and Chen [25]	Multiple	Multiple	Yes	—	Yes	Minimize the competition latency with energy consumption constraints.
Yadav et al. [26]	Multiple	Multiple	Yes	—	Yes	Minimize the energy consumption and delay in vehicular fog computing.
Yadav et al. [27]	Multiple	Multiple	Yes	—	—	Minimize the total energy consumption of the system with delay constraints.
This work	Multiple	Multiple	Yes	Yes	Yes	Minimize the average delay for SDN-enabled IoV system.

proposed in [20] for parallel computing and transmission of virtual reality. The task is divided into two subtasks and offloaded to the MEC server and the vehicle side in order to shorten the completion time of the virtual reality application. Moreover, an offloading scheme has been proposed with efficient privacy protection based on fog-assisted computing and solved by a joint optimization algorithm to minimize the completion delay in [24].

2.3. Optimizing Both System Delay and Energy Consumption. The user experience and the standby time of the device have been optimized together by the weight parameter, i.e., when the power is low, the weight of the energy consumption is set larger; when the power is sufficient, a larger delay weight is set. [19] consider a multicell wireless network that supports MEC, which assists mobile users perform computationally intensive tasks through task offloading. A heuristic algorithm is proposed to combine task offloading and resource allocation to maximize system utility, which is measured by the weighted sum of task completion time and energy consumption reduction. The system latency has been minimized with energy consumption constraints by jointly optimizing offloading decisions, local computing power, and fog node computing resources in [25]. The cloudlet overload risk has been alleviated by offloading user tasks to vehicle nodes, and a heuristic algorithm has been proposed to balance energy and delay to minimize system overhead in [26]. In order to solve the problem of high power consumption and delay sensitivity of portable devices, a reinforcement learning scheme has been proposed in [27] to search for optimal available resource nodes to minimize delay and energy consumption. In addition, the fog computing and cloud computing have been discussed in [24–27]. Since cloud servers are located in areas far away from cities, there is a large transmission delay, and tasks that are not sensitive to delay are offloaded to the cloud for computing, while intensive and delay-sensitive tasks are computed locally or offloaded to the fog to improve system performance.

In this paper, we mainly focus on delay-sensitive and task-intensive scenarios, considering that the computing resources of edge nodes and the number of tasks received in each period are limited. In order to prevent some edge computing nodes from being overloaded and some edge nodes relatively idle, we use the SDN-enabled IoV to control the task offloading decision-making by monitoring the global situation, which effectively improves the utilization of resources and reduce system latency.

3. System Model

In this section, the system transmission model, execution model, and optimization problem formulation are presented. As shown in Figure 1, we assume a vehicular network system is composed of N vehicles and M RSUs. Each RSU is equipped with a MEC server, which has the computing ability to process offloading tasks. Generally, the MEC can be a physical device or a virtual machine provided by the operator. Taking into account the complexity of the

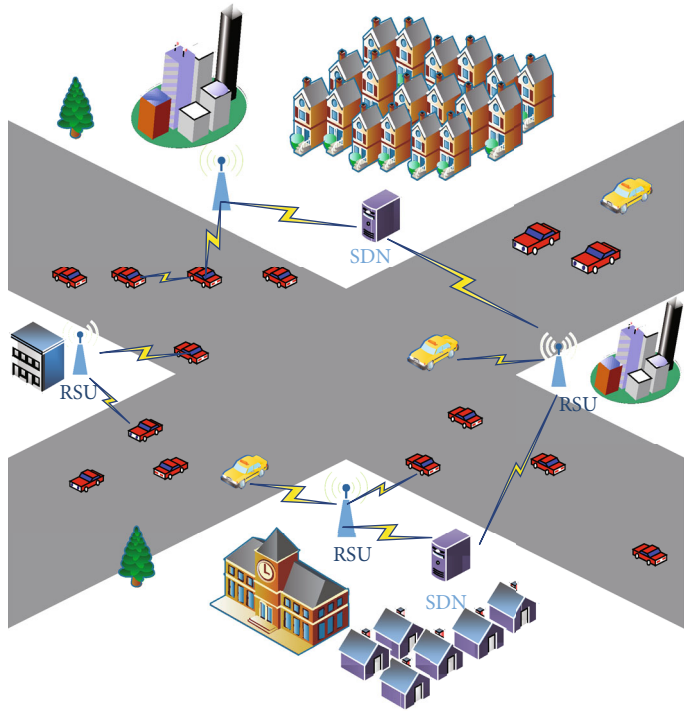

Figure 1: Sketch of communication and offloading framework for SDN-enabled IoV System.

vehicular network, this system is a software-defined IoV that supports edge computing and the configuration of edge computing nodes. The edge nodes are coordinated and controlled by the software-defined network (SDN), which aims to reduce system delay and improve overall performance. As a coordinator, SDN divides the IoV system into two independent layers through software definition and virtualization technology: the data layer and the control layer. The edge nodes uniformly obey SDN scheduling and follow the OpenFlow protocol [28]. These edge nodes transmit and process information according to SDN control instructions. The control and processing are separated by the network entities, to effectively integrate resources and improve utilization. The edge computing node equipped on RSU connects the edge node cluster and the SDN controller through broadband connection. The physical communication on the control layer is independent of the physical communication channel on the data layer. The data layer is composed of an OpenFlow-based SDN controller and network nodes, refers to [29, 30]. The SDN controller broadcast global status, including Channel Status Information (CSI), available resources, and task priority. When SDN receives the vehicle's offloading request, it looks for the best solution (including offloading decision and resource allocation) at the control layer and then sends control instructions. The data layer performs data transmission according to the received control layer. Each vehicle generates a task in a period of time, taking the heterogeneity of vehicle tasks into account (data volume, delay sensitivity, and difference in computational complexity). In the case of real-time tasks require minimal delay and/or have a large amount of data, local offloading cannot meet the requirements. Otherwise, tasks are not sensitive to delay response and can be processed locally. If all of them are executed locally or offloaded to the RSUs, it can cause timeout failure, waste of local resources, and a very poor communication environment due to interference. Therefore, different offloading strategies need to be set for different task types. Let $\mathscr{V} = \{1, 2, \cdots, N\}$ and $\mathscr{R} = \{1, 2, \cdots, M\}$ represent the set of vehicles and the set of RSUs, respectively. For ease of reference, the key symbols used in this article are summarized in Table 2.

3.1. Communication Model. We assume that each vehicle terminal $n \in \mathscr{V}$ has an executed task at a time and denoted as V_n. Each task has three parameters, $\langle d_n, c_n, t_n^{\max} \rangle$, in which d_n defines the size of the input data of the task V_n of the vehicle terminal n (usually in bits), and c_n defines the computing resources required by the task of the terminal n, which refer to CPU cycles. Parameters d_n and c_n can be obtained from task analysis. $t_n^{\max}$ is the maximum allowable delay for task transmission execution, i.e., if $t_n^{\max}$ is exceeded by the time of result received, and the task is failed by timeout. Since the vehicles receive the offloading decision, the tasks are not allowed to be interrupted before the execution is completed. Typically, the speed of cars on conventional road is 5 to 16 meters per second; thus, we assume the radio channel is not radical varying to interrupt the execution of the tasks, due to the severe fading. When the vehicle generates a task, it sends an offloading request to the nearby RSU first; then, the RSU routes the request command to the SDN controller. SDN synthesizes the current global information and provides the optimal offloading decision. Finally, the decision plan is sent to the targeted node through the control layer.

Table 2: Parameter notations.

Symbol	Definition
d_n	The input data size of task V_t generated by vehicle n.
B^{RV}	The bandwidth of vehicle to RSU channel.
B^{RR}	The bandwidth of RSU to RSU.
c_n	The computing resources required by the vehicle's task.
C_n^m	Indicator of the selection on RSU m by the vehicle n.
$t_n^{\max}$	The maximum allowable delay for task execution.
h_n^s	The uplink gain of the user n to routing RSU m channel.
l_n	The computing resource of the vehicle's task.
b_n	The proportion of local execution in the task.
r_n^s	The transmission rate from vehicle to routing RSU.
r_s^m	The transmission rate from routing RSU to targeted RSU.
T_n^{loc}	The local task execution time.
f_m	The computing resource of the RSU server m.
f_n^m	The computing resources allocated by the RSU m to vehicle n.
$T_n^{m,up}$	The transmission time for the vehicle n to upload.
T_n^s	The transmission time from the vehicle n to routing RSU s.
T_s^m	The transmission time from routing RSU s to targeted RSU m.
$T_n^{m,comp}$	The execution time.
T_n	The delay of vehicle n completing task.
ϕ	The computing resources (CPU cycles) for 1 bit processing.
ζ_n^m	Indicator of routing decision from the vehicle n to the targeted RSU m.
$\aleph$	The path loss index.
h_0	The complex Gaussian channel coefficient.
x_i	The position of particle.
(x_n, y_n)	The coordinates of vehicle.
(X_m, Y_m)	The coordinates of RSU.
Ξ_n^m	The distance of vehicle and RSU.
σ^2	The additive white Gaussian noise of the channel.

We introduce a binary variable of task offloading selection C_n^m. If C_n^m =1, it defines that the RSU m is selected by vehicle n to perform task offloading. On the contrary, C_n^m =0 means the vehicle n do not select the RSU m. Therefore, the constraint of C_n^m is defined as

$$\sum_{m \in R} C_n^m = 1, \quad \forall n \in \mathscr{V}. \tag{1}$$

Equation (1) means that each vehicle can offload tasks to a unique RSU for execution. We define the coordinates of the RSUs set in two-dimensional plane as $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_M, Y_M)\}$, the coordinates of the vehicle n are defined as (x_n, y_n), and the coordinates of nearest RSU s are denoted by (X_s, Y_s). Then the distance between the vehicle n and the nearest RSU s is given by

$$\Xi_n^m = \sqrt{(x_n - X_s)^2 + (y_n - Y_s)^2}. \tag{2}$$

The transmission rates of vehicle n to routing RSU s channel and routing RSU s to target RSU m channel are described as [31]

$$\begin{cases} r_n^s = B^{RV} \log_2 \left(1 + \dfrac{h_n^s (\Xi_n^s)^{-\aleph} |h_0|^2}{\sigma^2}\right), \\ r_s^m = B^{RR} \log_2 \left(1 + \dfrac{h_s^m (\Xi_s^m)^{-\aleph} |h_0|^2}{\sigma^2}\right), \end{cases} \tag{3}$$

where B^{RV} and B^{RR} represent the bandwidth allocations of vehicle n to routing RSU s channel and routing RSU s to target RSU m, respectively. σ^2 represents additive white Gaussian noise of the channel. h_n^s represents the transmission power from routing RSU s to vehicle n. h_0 is the complex Gaussian channel coefficient [31] following the complex normal distribution CN (0,1). $\aleph$ is the path loss index.

We introduce b_n to represent the local execution ratio of the vehicle; thus, $(1 - b_n)$ denotes the offload ratio. Since the execution result is usually relatively small, the delay from targeted node to the routing node and the delay from the targeted node can be ignored. We define the binary parameter ζ_n^m, where ζ_n^m is 1 means the vehicle needs to be routed to the targeted node, otherwise, vice versa. Assuming the routing node is RSU s, based on the above formulas, the transmission time for the uploading $(1 - b_n)d_n$ data from vehicle to the targeted RSU m is divided into two parts: (1) the delay of uploading from the vehicle to the routing node and (2) the delay of uploading from the routing node to the targeted node, which are expressed as

$$\begin{cases} T_n^s = \dfrac{(1 - b_n)d_n}{r_n^s}, \quad \forall n \in \mathscr{V}, \\ T_s^m = \dfrac{(1 - b_n)d_n}{r_n^m}, \quad \forall n \in \mathscr{V}. \end{cases} \tag{4}$$

Considering that the routing node can be the candidate of targeted node, the uploading time $T_n^{m,up}$ is defined as

$$T_n^{m,up} = \zeta_n^m (T_n^s + T_s^m) + \left(1 - \zeta_n^m\right) T_n^s, \forall n \in \mathscr{V}. \tag{5}$$

3.2. Computing Model. In this section, we introduce the computing model of the vehicle. Considering the parallel offloading mode, the computing model is mainly divided into two parts: (1) the local computing model and (2) RSU computing model.

3.2.1. Local Computing Model. Let l_n denotes the computing capabilities of vehicle n; then, the local task execution time T_n^{loc} on b_n data can be defined as

$$T_n^{loc} = \frac{b_n c_n}{l_n}, \quad \forall n \in \mathscr{V}. \tag{6}$$

3.2.2. RSU Computing Model. Besides the local computing, the rest part of the tasks are offloaded to the RSU for further computing. It requires (1 - b_n) c_n computing resources from RSU. Note that a RSU can be selected by multiple vehicles, and a vehicle can only select one RSU for execution. Since RSU has limited computing resources, it is necessary to allocate RSU resources to different vehicles in a reasonable manner to improve resource utilization and reduce system delay. Define f_m as the computing resource of RSU m, and f_n^m denotes the computing resources allocated by the RSU server m to vehicle n. The sum of the resources allocated by the RSU to each vehicle cannot exceed its own computing resources, thus constrained by Equations (7) and (8), and the execution time of vehicle n on RSU m is defined by Equation (9).

$$\sum_{n \in V} f_n^m \le f_m, \quad \forall m \in \mathscr{R}, \tag{7}$$

$$\frac{f_n^m}{f_m} \le C_n^m, \quad \forall m \in \mathscr{R}, \tag{8}$$

$$T_n^{m,comp} = \frac{C_n^m (1 - b_n) c_n}{f_n^m}, \quad \forall n \in \mathscr{V}. \tag{9}$$

Assuming there exist M RSUs in the network, then the execution time for vehicle n is defined as

$$T_n^{RSU} = \sum_{m=1}^{M} \frac{C_n^m (1 - b_n) c_n}{f_n^m}, \quad \forall n \in \mathscr{V}. \tag{10}$$

For vehicle n, the task is divided into two subtasks, which are processed in parallel by local unit and by edge RSU node. The task completion time is accordingly divided into two parts: local processing time and edge RSU node processing time. The edge RSU node processing also results transmission delay $T_n^{m,up}$ and execution delay $T_n^{m,comp}$. Therefore, the time for completing task generated by vehicle n is determined by the maximum delay, which is defined by

$$T_n = \max \left\{ T_n^{loc}, \sum_{m=1}^{M} (T_n^{m,comp} + T_n^{m,up}) \right\}, \quad \forall n \in \mathscr{V}. \tag{11}$$

3.3. Problem Formulation. For delay-sensitive applications, delay is a problem that must be considered. Therefore, it is vital to formulate the most suitable offloading strategy based on the global status. For intensive tasks, meeting the performance requirements of the vehicle and making full use of effective resources are of importance. In addition, considering the transmission and execution delay caused by offloading, the system delay is defined by Equation (12). Therefore, offloading decision-making and resource allocation must be jointly optimized to improve system performance. The goal is to provide all vehicle optimized offloading strategies χ, computing resource allocation $\mathscr{F}$, and offloading ratio $\mathscr{B}$, aiming to reduce average delay. Finally, the optimization problem is described as follows:

$$P1 : D_n = \min_{\mathscr{B},\chi,\mathscr{F}} \frac{1}{N} \sum_{n=1}^{N} T_n \tag{12}$$

$$s.t. T_n \le t_n^{\max}, \quad \forall n \in \mathscr{V} \tag{13a}$$

$$\sum_{m \in R} C_n^m = 1, \quad \forall n \in \mathscr{V} \tag{13b}$$

$$0 \le b_n \le 1, \quad \forall n \in \mathscr{V} \tag{13c}$$

$$C_n^m \in \{0, 1\}, \quad \forall n \in \mathscr{V}, \forall m \in \mathscr{R} \tag{13d}$$

$$0 \le \sum_{n=1}^{N} C_n^m f_n^m \le f_m, \quad \forall m \in \mathscr{R} \tag{13e}$$

$$\zeta_n^m \in \{0, 1\}, \quad \forall n \in \mathscr{V}, \forall m \in \mathscr{R}. \tag{13f}$$

The above constraints are explained as follows: constraint (13a) is to ensure that the total delay of the task does not exceed the maximum allowable delay; constraints (13b) and (13d) mean that each vehicle can only transmit the task to only one RSU, and the offloading decision is a binary variable; constraint (13c) is the offloading ratio constraint, which is a decimal between 0 and 1; and to ensure that the resources allocated by the RSU to the vehicle does not exceed its own capacity, the constraint (13e) must be met. Finally, there is a binary constraint, which represents whether the vehicle to the targeted RSU needs to be routed.

4. Problem Solving

The fact that the offloading decision is a binary variable brings problem $P1$ as a mixed integer programming problem, which is nonconvex NP-hard. In order to reduce the complexity of the problem, we divide the problem into three subproblems, i.e., offloading decisions of various vehicles, proportional distribution of tasks performed locally, and different resource allocation for vehicles by RSUs.

4.1. Offloading Strategy Making and Load Balance. Given the local computing ratio $\mathscr{B}$ and resource allocation $\mathscr{F}$, problem $P1$ is transformed into $P1.1$ as follows:

$$P1.1 : D_n = \min_{\chi} \frac{1}{N} \sum_{n=1}^{N} T_n \tag{14}$$

$$s.t. (12a), (12b), (12d), (12e), (12f). \tag{15}$$

Obviously, $P1.1$ is still an NP-hard problem. For such problems, heuristic algorithm is a rational solution. The Particle Swarm Optimization algorithm (PSO) originated from the study of bird predation behavior is a evolutionary heuristic algorithm, which has efficient global search capabilities. It has achieved great success in image processing and neural network training. Hence, we propose a offloading decision-making algorithm based on PSO. The basic idea of PSO is to find the optimal solution through collaboration and information sharing between individuals in the group. In the paper, particles have only two attributes: speed and position. Speed represents the speed of movement, and position represents the direction of particles. Each particle searches for the optimal solution separately in the search space and records the current individual extreme value, then sharing the individual extreme value with other particles. All particles in the particle swarm adjust their speed and position according to the current individual extreme value they find and the current global optimal solution shared by others. Therefore, through iteration, the particles of the entire population eventually converge to the optimal solution.

According to the speed update formula proposed by Clerc and Kennedy [32], we introduced a compression factor x. The update formula for speed and position is presented as

$$\begin{aligned} V_i(I+1) = \rho(wV_i(I) + c_1 r_1(Pbest - X_i(I)) \\ + c_2 r_2(Gbest - X_i(I))), \end{aligned} \tag{16}$$

$$X_i(I+1) = X_i(I) + V_i(I), \tag{17}$$

where $i = 1, 2, \cdots Nc$ max., Nc max is the number of particles. w is the coefficient of inertia. The larger w, the stronger the global optimization ability, and the weaker the local optimization ability. c_1, c_2 are the learning factors, $V_i(I)$ is the velocity of particle i at the $I-th$ iteration, and $X_i(I)$ is the current position of particle i at the $I-th$ iteration. $V_{\max}$ denotes the maximum velocity.

The compression factor ρ is given by

$$\rho = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}, \tag{18}$$

where $\varphi = c_1 + c_2$. The compression factor guarantees the convergence of the particles and prevents the explosion of velocity. The specific process of the improved PSO algorithm is described in Algorithm 1. The $fitness$ function refers to Equation (12).

4.2. Resource Allocation Optimization. Assuming the local computing ratio $\mathscr{B}$ and offloading decision χ are given, the problem is transformed into the lowest vehicle delay to each RSU. Defining the vehicle set $\mathcal{N}_m$ of the task on RSU m, and the number of vehicles is N_m. The variable f_n^m only has an impact on the task execution of the objective function; thus, the converted problem $P1.2$ can be presented as

$$\begin{aligned} P1.2 : \min_{\mathscr{F}} \sum_{n \in \mathcal{N}_m} T_n \\ s.t. (12a), (12e). \end{aligned} \tag{19}$$

Substituting formulas (4), (5), and (6) into $P1.2$, and performing equivalent transformation, the overall optimization problem is changed to the following form:

$$\begin{aligned} P1.3 : \min_{\mathscr{F}} \sum_{n \in \mathcal{N}_m} T_n \\ s.t. T_n \geq \frac{b_n d_n}{l_n} \\ T_n \geq \left(\frac{(1-b_n)d_n}{r_n^s} + \frac{\zeta_n^m (1-b_n) d_n}{r_s^m} + \frac{(1-b_n)c_n}{f_n^m} \right) \\ (12a), (12e). \end{aligned} \tag{20}$$

As $P1.3$ shows, T_n is not differentiable subject to f_n^m. Substituting formulas (5), (6), and (9) into $P1.3$, then the T_n is approximated as

$$\begin{aligned} T_n &\leq \frac{b_n c_n}{l_n} + \frac{(1-b_n)d_n}{r_n^s} + \frac{\zeta_n^m (1-b_n) d_n}{r_n^m} + \frac{(1-b_n)c_n}{f_n^m} \\ &= \frac{(1-b_n)c_n}{f_n^m} + b_n \left(\frac{c_n}{l_n} - \frac{d_n}{r_n^s} - \frac{\zeta_n^m d_n}{r_n^m} \right) + \Gamma_n^m, \end{aligned} \tag{21}$$

where $\Gamma_n^m = d_n / r_n^s + \zeta_n^m d_n / r_n^m$ and $(1-b_n)c_n / f_n^m + b_n((c_n/l_n) - (d_n/r_n^s) - (\zeta_n^m d_n / r_n^m)) + \Gamma_n^m$ is the upper bound of T_n. Substitute the upper bound into $P1.3$, then $P1.3$ can be bounded with the worst case delay as

$$P1.4 : \min_{\mathscr{F}} \sum_{n \in \mathcal{N}_m} \frac{(1-b_n)c_n}{f_n^m} + b_n \left(\frac{c_n}{l_n} - \frac{d_n}{r_n^s} - \frac{\zeta_n^m d_n}{r_n^m} \right) + \Gamma_n^m \tag{22}$$

$$s.t. \frac{(1-b_n)c_n}{f_n^m} + b_n \left(\frac{c_n}{l_n} - \frac{d_n}{r_n^s} - \frac{\zeta_n^m d_n}{r_n^m} \right) + \Gamma_n^m \leq t_n^{\max} \tag{23a}$$

$$(12e). \tag{24}$$

$P1.4$ is a convex optimization problem, with linear constraints (13e) and convex inequality constraints (23a). Therefore, we use Lagrangian duality theory to solve this problem. The Lagrangian function of $P1.4$ is given by

$$\begin{aligned} L(f, \lambda, \theta) = \sum_{n=1}^{N_m} \left(\frac{(1-b_n)c_n}{f_n^m} + b_n \left(\frac{c_n}{l_n} - \frac{d_n}{r_n^s} - \frac{\zeta_n^m d_n}{r_n^m} \right) + \Gamma_n^m \right) \\ + \sum_{n=1}^{N_m} \lambda_n \left(\frac{(1-b_n)c_n}{f_n^m} + \tau \right) + \theta_m \left(\sum_{n \in \mathcal{N}_m} f_n^m - f_m \right), \end{aligned} \tag{25}$$

Input:
The input parameters of particle swarm:
$c_1, c_2, w, r_1, r_2, Mcmax, Ncmax$.
Output:
$Gbest, \ominus Gbest$
1: **For** $j = 1; j <= Mcmax; j++$ **do**
2: **For** each $i \in [1, Ncmax]$ **do**
3: Initialize the velocity position of particles $V_i(0), X_i(0)$
4: **End for**
5: In Algorithm 2, we obtain ζ_n^m then record the current position and fitness as particle's individual extreme option and value $Pbest, \ominus Pbest$.
6: Record the smallest fitness and the corresponding position as $\ominus Gbest, Gbest$.
7: **While** the number of iteration steps is not 0 **do**
8: **For** $i = 1; i <= Ncmax; i++$ **do**
9: Update the velocity $V_i(j)$ of particles i using Equation (16)
Update the position $X_i(j)$ of i using Equation (17)
Evaluate fitness of particle i using Equation (14)
10: **If** $fitness(X_i(j)) < \ominus Pbest$ **then**
11: $\ominus Pbest = fitness(X_i(j))$
12: **End if**
13: **If** $fitness(X_i(j)) < \ominus Gbest$ **then**
14: $\ominus Gbest = fitness(X_i(j))$
15: **End if**
16: **End for**
17: End **while**
18: **End for**
19: **Return** $Gbest, \ominus Gbest$

Algorithm 1: A offloading decision making algorithm based on PSO.

Input:
Offloading decison: C_n^m.
Output:
Routing information: ζ_n^m.
1: Initialize the location of the vehicles.
2: **For** each $i \in [1, N]$ **do**
3: Obtain the access RSU m of vehicle i, set $\zeta_n^m = 0$.
4: **For** $j = 1; j <= M; j++$ **do**
5: **If** $C_i^j = 1$ and $j! = m$ **then**
6: $\zeta_n^m = 1$.
7: **End if**
8: **End for**
9: **End for**
10: **Return** ζ_n^m

Algorithm 2: Routing confirmation (RC).

where $\tau = b_n((c_n/l_n) - (d_n/r_n^s) - (\zeta_n^m d_n/r_n^m)) + \Gamma_n^m - t_n^{\max}$, where λ_n and θ_m are the Lagrangian multipliers. According to KKT conditions, we have

$$f_n^m = \left[\left[\sqrt{\frac{(1+\lambda_n)(1-b_n)c_n}{\theta_m}}\right]^+\right]. \tag{26}$$

The Lagrange dual function is then given by

$$\mathscr{D}(\lambda, \theta) = \min\ L(f, \lambda, \theta). \tag{27}$$

Then, the dual problem of $P1.4$ is

$$\begin{aligned} &\max\ \mathscr{D}(\lambda, \theta) \\ &s.t. \lambda_n, \theta_m \geq 0. \end{aligned} \tag{28}$$

As the Lagrange function is differentiable, the gradients of the Lagrange multipliers can be obtained by

$$\begin{aligned} \frac{\partial \mathscr{D}(\lambda, \theta)}{\partial \lambda_n} &= \frac{(1-b_n)c_n}{f_n^m} + \tau, \\ \frac{\partial \mathscr{D}(\lambda, \theta)}{\partial \theta_m} &= \sum_{n \in \mathscr{N}_m} f_n^m - f_m, \end{aligned} \tag{29}$$

The Lagrange multiplier iterative formula is as follows by using gradient descent.

$$\lambda_n(t+1) = \left[\lambda_n(t) - \eta_1 \frac{\partial \mathscr{D}}{\partial \lambda_n}\right]^+, \tag{30}$$

$$\theta_m(t+1) = \left[\theta_m(t) - \eta_2 \frac{\partial \mathscr{D}}{\partial \theta_m}\right]^+, \tag{31}$$

where η_1, η_2 are the gradient steps, t represents gradient number, and $[.]^+$ represents $\max(0, .)$. We summarize the procedure for solving problem $P1.4$ in Algorithm 3.

4.3. Offloading Radio Allocation. Given χ and $\mathscr{F}$, the $P1.4$ problem can be transformed into a linear programming

Input:
λ_n, θ_m
Output:
f_n^m
1: Repeat
2: Calculate resource allocation f_n^m based on Equation (26)
3: Update the $\lambda_n(t), \theta_m(t)$ using Equation (30)
4: Until convergence
5: **Return** f_n^m

Algorithm 3: Resource allocation (RA).

1: Set offloading decision X^0, resource allocation F^0, and local executing ratio B^0
2: $t = 0$
3: **While** t<T **do**
4: Obtain the offloading decision by Algorithm 1 based on F^{t-1}, B^{t-1}
5: Calculate resource allocation by Algorithm 3 based on B^{t-1}, X^t
6: Update the B^t by using Equation (13c), (23a) based on F^t, X^t
7: $t = t + 1$
8: **End while**
9: **Return**B^t, X^t, F^t

Algorithm 4: Joint approach for offloading decision, resource allocation and ratio (ODRR).

problem on b_n. Therefore, the function value takes the extreme value at the boundary or stagnation point, as shown in $P1.5$

$$P1.5: \mathscr{W} = \min_{\mathscr{B}} \sum_{n \in \mathscr{N}_m} \frac{(1-b_n)c_n}{f_n^m} + b_n\left(\frac{c_n}{l_n} - \frac{d_n}{r_n^s} - \frac{\zeta_n^m d_n}{r_n^m}\right) + \sum_{n \in \mathscr{N}_m} \Gamma_n^m \quad s.t.(12c), (20a). \tag{32}$$

Based on the above formula, $\partial\mathscr{W}/\partial b_n$ as a constant and the minimum value can be obtained at the boundary. According to the constraints (13c) and (23a), b_n can be obtained.

Taken together, we propose an approach combining the whole process of offloading strategy, offloading ratio, and resource allocation control (ODRR), as described in Algorithm 4.

5. Numerical Results

In this section, simulation configurations and results are presented and analyzed to verify the effectiveness of the proposed algorithm.

5.1. Simulation Configurations. The scenario in this paper is as follows: the system consists of 3 RSUs and N vehicles $(N = 10,20,30\cdots)$, which carry tasks with random parameters, different data volumes, computing resources, and allowable delays. Assuming that the computing resources of 3 RSUs are $[15G, 15G, 15G]$, the local resources of the vehicle are $1G$. Vehicles' task data volume, computing complexity, and maximum allowable delay are random $(100KB, 300KB)$, random $(1000, 9000)$, random $(1s, 2s)$. Referring to [31, 33, 34], communication parameter settings are shown in Table 3. The settings of the parameters in Algorithm 1 are as follows: the number of particles is 100, the maximum number of iterations is 50, the learning factors c_1 and c_2 are both equal to 2, because the learning factors are parameters for adjusting the step size. If the setting is too large, the particle moves fast and fly over the optimal point. If set small, the optimization speed will be slow. Larger inertia weight has stronger global search ability and slower convergence speed. In order to avoid falling into the local optimal solution and have a faster convergence speed, it is most appropriate to set ω to 0.9. In order to verify the effectiveness

Table 3: Simulation configurations.

Parameter	Value
The bandwidth B^{RV},B^{RR}	1 MHZ
The path loss index $(\aleph)$	4
The additive white Gaussian noise (σ^2)	-100 dBm
h_n^s	20 dBm
h_s^m	46 dBm
The learning factors c_1, c_2	2
The inertia weight ω	0.9

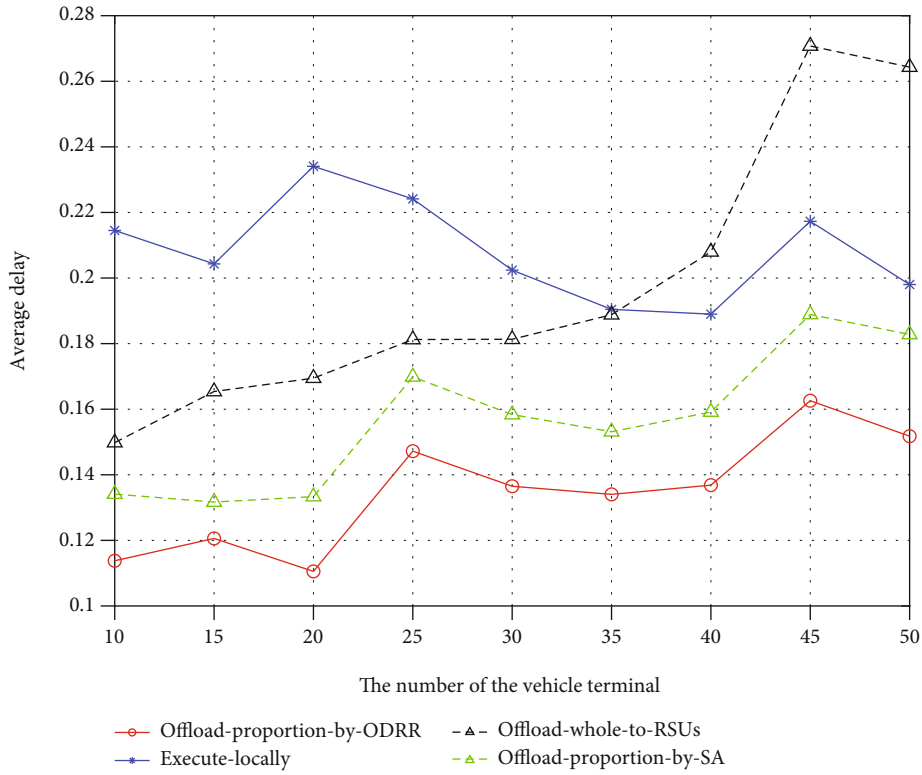

Figure 2: The system delay of the cases with different vehicle numbers ($N = 10, 20, 30, \cdots 50$).

of the proposed offloading strategy on the proposed Internet of Vehicles architecture, we compare it with other offloading strategies.

(i) Offload-Whole-to-RSUs: offload the whole task to edge nodes including access node and remote nodes

(ii) Execute-Locally: the vehicle terminal directly executes the task locally

(iii) Offload-proportion-by-ODRR: joint optimization of offloading decision, local calculation ratio, and resource allocation (ODRR) is based on the proposed algorithm

(iv) Offload-proportion-by-SA: offload proportion of task to RSUs using simulated annealing algorithm (SA).

5.2. Simulation Results. In this section, we present the performance of the proposed ODRR algorithm and compare with other conventional offloading strategies.

Figure 2 plots the average execution time of vehicles as the number of connected vehicles increases. It can be seen that the performance of the proposed ODDR partial offloading algorithm is the best and can keep the average calculation delay to a minimum, and the partial offloading SA algorithm is behind it. When the number of vehicles is less than 35, the effect of offloading whole to RSUs is better than executing locally. Because RSUs have more computing resources than the vehicles, the computing capability of RSU is dozens of times that of executing locally. Therefore, computing performance of executing locally is worse than offloading whole to RSUs. But when the number exceeds the limit, the situation becomes different. On the one hand, this is because the resources of RSUs are limited. When the number of offloading vehicles exceeds a certain number, the load capacity of RSUs is exceeded, resulting in great performance decrease. On the other hand, more vehicles means a worse communication environment, which leads to more communication delays. Compared with the other three conventional strategies, the latency of the proposed ODRR algorithm is always the smallest, as the number of vehicles increases. Compared with the Offload-Whole-to-RSUs scheme, the ODRR can be reduced by 42.7% in the best case, and 17.5% in the worst case; compared with the Execute-Locally scheme, the ODRR can be reduced by 52.6% in the best case, and 24% in the worst case; compared with Offload -proportion-by-SA scheme, the highest can be reduced by 16.7%, and the lowest by 7.8%. It can be concluded that compared with other two conventional strategies, ODRR can reduce the delay by up to nearly half. Compared to the Offload -proportion-by-SA scheme, the reduction is also up to 10%.

Figure 3 plots the impact of different executing complexities ($\phi = c_n/d_n$) on system delay. The number of connected vehicles is set to 30, and it is found that the average delay of the four offloading strategies increases linearly with the increase of task complexity. The performance of the ODRR algorithm given in this paper is obviously the best compared with other strategies. The partial offloading by SA algorithm is the second, the whole offloading to RSUs is the third, and

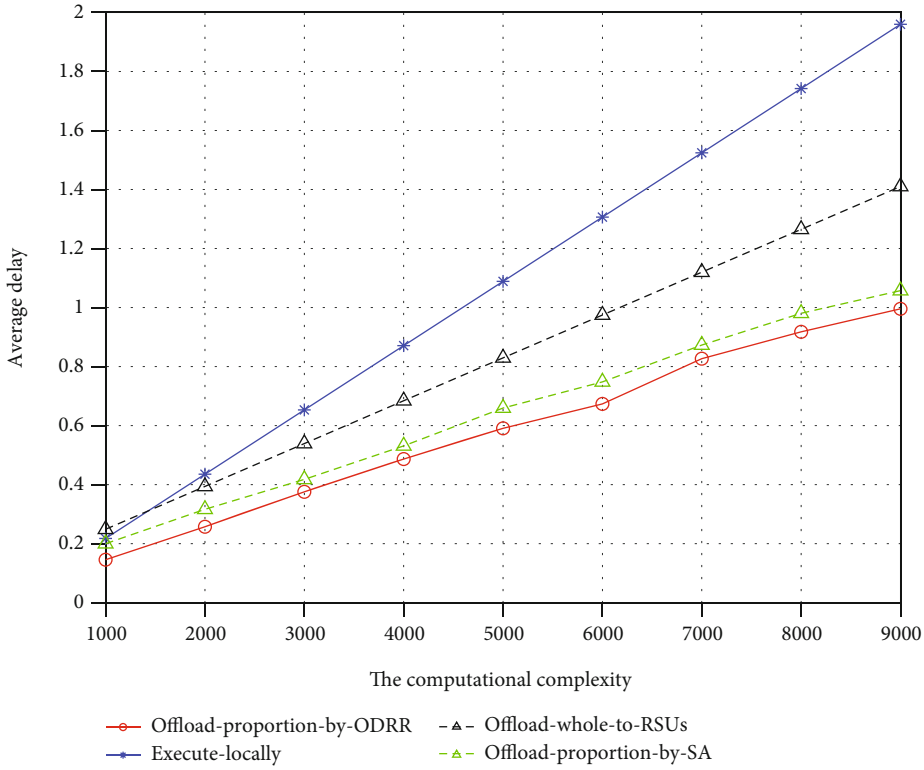

Figure 3: The average delay of the cases with different execution complexities.

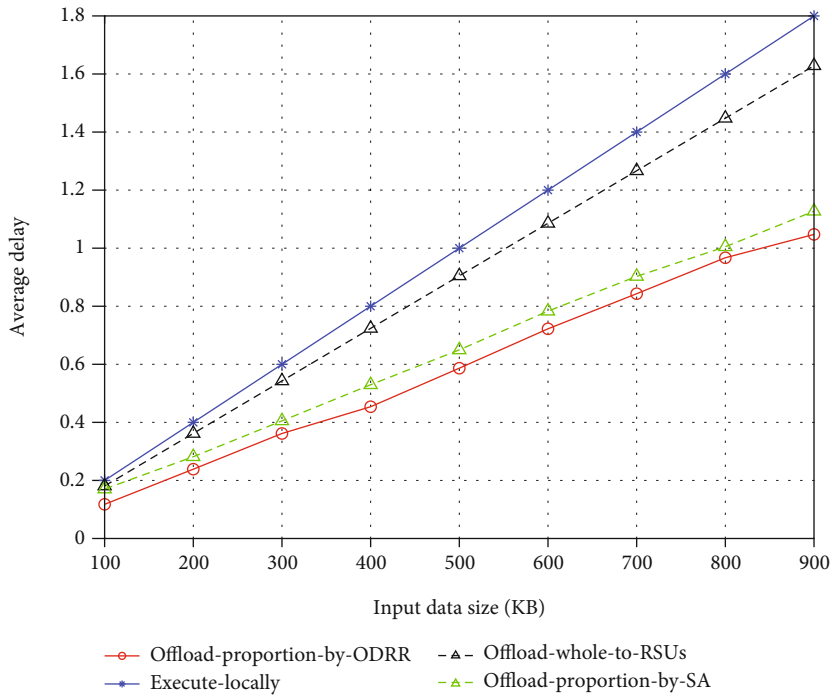

Figure 4: The average delay of the cases with different execution input data size.

executing locally has the worst performance. The higher the task complexity, the more CPU resources are needed to process each byte of data. The local computing resources are much smaller than RSUs, so executing locally gets the highest latency. However, offloading whole to RSUs causes uneven distribution and local resource waste. The partial offloading ODRR algorithm takes into account both resource allocation and offloading ratio, which greatly improves

system performance. Compared with the Offload-Whole-to-RSUs scheme, the ODRR can be reduced ranging from 14% to 29%; compared with the Execute-Locally scheme, the ODRR can be reduced ranging from 14% to 49.7%, and when the computational complexity is greater, the reduction effect is better. Compared with the Offload-proportion-by-SA scheme, the reduction is between 7% and 9%.

When $N = 30, \phi = 2000$, Figure 4 shows the change of average delay with the amount of input data increasing. The average execution delay of the vehicles increases linearly with the size of the input data increasing. The proposed ODRR algorithm obtains the minimum delay, after the SA algorithm, whole offloading to RSUs is the third, and only executing locally is final. The larger the input data, the more transmission delay is caused. The algorithm proposed in this paper jointly optimizes the offloading ratio, offloading decision-making, and resource allocation, thus greatly improving the system performance. With the larger amount of input data, the ODRR proposed in this paper has a better effect on reducing the delay. Compared with Execute-Locally scheme, the reduction is between 36.6% and 47.5%. Compared with Offload-Whole-to-RSUs scheme, it can reduce the delay by 30.4% to 42%; compared with the Offload-proportion-by-SA scheme, the decline rate is nearly 10%.

6. Conclusion

In this paper, we propose a multiuser and multi-RSU system architecture based on SDN-enabled IoV. The loads of RSUs are effectively balanced by using the characteristics of SDN. In order to reduce the delay of task offloading in IoV, a joint approach is proposed to optimize the offloading ratio, offloading decision-making, and resource allocation. Compared with the conventional work that executing locally, the system performance increases 36.6% – 47.5%. Compared with the decision by fully offloading to RSUs, the performance increases 30.4% – 42%; compared with the offloading strategy by SA, the performance increases about 10%. The simulation results have greatly proved that the joint optimization approach proposed in this paper is more effective than conventional strategies in dealing with the delay problem of multiuser and multi-RSU system and can effectively solve the multidimensional problem. Although greatly improving system performance, there is still room for improvement. For instance, the possibility of task failure caused by transmission link or edge node failure has not considered in this work. Therefore, the reinforcement learning-based approach is of interest to solve the optimization goal considering the failure retransmission mechanism of the task.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under grants No. 61901104 and the Science and Technology Research Project of Shanghai Songjiang District No. 20SJKJGG4 (Corresponding author: Lei Zhang).

References

[1] J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in internet of vehicles: a review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2339–2352, 2015.

[2] M. Ashritha and C. S. Sridhar, "RSU based efficient vehicle authentication mechanism for VANETs," in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1–5, Coimbatore, India, 2015.

[3] Z. Wang, J. Zheng, Y. Wu, and N. Mitton, "A centrality-based RSU deployment approach for vehicular ad hoc networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–5, Paris, France, 2017,.

[4] M. Bahrami, "Cloud computing for emerging mobile cloud Apps," in *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pp. 4-5, San Francisco, CA, USA, 2015.

[5] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-oriented MEC applications placement in a federated edge cloud architecture," in *ICC 2020-2020 IEEE international conference on communications (ICC)*, pp. 1–6, Dublin, Ireland, 2020.

[6] S. Lee, S. Lee, and M.-K. Shin, "Low cost MEC server placement and association in 5G networks," in *2019 International conference on information and communication technology convergence (ICTC)*, pp. 879–882, Jeju, Korea (South), 2019.

[7] Y. Wang, J. Wang, Y. Ge, B. Yu, C. Li, and L. Li, "MEC support for C-V2X system architecture," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pp. 1375–1379, Xi'an, China, 2019.

[8] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing–a key technology towards 5G," *ETSI White Paper*, vol. 11, 2015.

[9] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[10] J. Du, L. Zhao, X. Chu, F. R. Yu, J. Feng, and I. Chih-Lin, "Enabling low-latency applications in LTE-A based mixed fog/cloud computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1757–1771, 2019.

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

[12] H. Wang, Z. Lin, and T. Lv, "Energy and delay minimization of partial computing offloading for D2D-assisted MEC systems," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, Nanjing, China, 2021.

[13] J. Wang, T. Lv, P. Huang, and P. T. Mathiopoulos, "Mobility-aware partial computation offloading in vehicular networks: a deep reinforcement learning based scheme," *China Communications*, vol. 17, no. 10, pp. 31–49, 2020.

[14] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[15] H. Lee, H. Kim, and Y. Kim, "A practical SDN-based data offloading framework," in *2017 International Conference on Information Networking (ICOIN)*, pp. 604–607, Da Nang, Vietnam, 2017.

[16] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Communications*, vol. 17, no. 5, pp. 266–283, 2020.

[17] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[18] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, 2019.

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

[20] J. Zhou, F. Wu, K. Zhang, Y. Mao, and S. Leng, "Joint optimization of offloading and resource allocation in vehicular networks with mobile edge computing," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, Hangzhou, China, 2018.

[21] K. Zhang, Y. Mao, S. Leng et al., "Energy-Efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[22] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[23] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-Optimal dynamic computation offloading for industrial IoT in fog computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2020.

[24] S. Chen, X. Zhu, H. Zhang, C. Zhao, G. Yang, and K. Wang, "Efficient privacy preserving data collection and computation offloading for fog-assisted IoT," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 4, pp. 526–540, 2020.

[25] Q. Wang and S. Chen, "Latency–minimum Offloading Decision and Resource Allocation for Fog enabled Internet of Things Networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 12, 2020.

[26] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14198–14211, 2020.

[27] R. Yadav, W. Zhang, I. A. Elgendy et al., "Smart healthcare: RL-based task offloading scheme for edge-enable sensor networks," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24910–24918, 2021.

[28] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: from concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[29] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 94–100, 2017.

[30] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: architecture and services," in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pp. 103–110, Piran, Slovenia, 2014.

[31] D. Ye, M. Wu, S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 247–251, Beijing, China, 2016.

[32] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

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

[34] Y. Qi, H. Wang, L. Zhang, and B. Wang, "Optimal access mode selection and resource allocation for cellular-VANET heterogeneous networks," *IET Communications*, vol. 11, no. 13, pp. 2012–2019, 2017.