A Context-aware Task Offloading Scheme in Collaborative Vehicular Edge Computing Systems

Zilong Jin[1,2], Chengbo Zhang[1], Guanzhe Zhao[3*], Yuanfeng Jin[4], and Lejun Zhang[5]
[1] School of Computer and Software, Nanjing University of Information Science and Technology
Nanjing 210044, China
[e-mail: zljin@nuist.edu.cn, zcbnuist@gmail.com]
[2] Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET)
Nanjing University of Information Science and Technology, Nanjing 210044, China
[3] Huihua College of Hebei Normal University,Shijiazhuang 050091, China
[e-mail: GuanzheZhao@outlook.com]
[4] Department of Physics, Yanbian University, Yanji 133002, China
[e-mail: jinyuanfeng1976@hotmail.com]
[5] College of Information Engineering, Yangzhou University, Yangzhou 225127, China
[e-mail: zhanglejun@yzu.edu.cn]
*Corresponding author: Guanzhe Zhao

Received December 11, 2020; revised January 11, 2021; accepted January 23, 2021; published February 28, 2021

Abstract

With the development of mobile edge computing (MEC), some late-model application technologies, such as self-driving, augmented reality (AR) and traffic perception, emerge as the times require. Nevertheless, the high-latency and low-reliability of the traditional cloud computing solutions are difficult to meet the requirement of growing smart cars (SCs) with computing-intensive applications. Hence, this paper studies an efficient offloading decision and resource allocation scheme in collaborative vehicular edge computing networks with multiple SCs and multiple MEC servers to reduce latency. To solve this problem with effect, we propose a context-aware offloading strategy based on differential evolution algorithm (DE) by considering vehicle mobility, roadside units (RSUs) coverage, vehicle priority. On this basis, an autoregressive integrated moving average (ARIMA) model is employed to predict idle computing resources according to the base station traffic in different periods. Simulation results demonstrate that the practical performance of the context-aware vehicular task offloading (CAVTO) optimization scheme could reduce the system delay significantly.

Keywords: Differential Evolution, Mobile Edge Computing, Machine Learning, Computing Offloading, Context-aware

This work was partially supported by the National Natural Science Foundation of China (Grant No.61602252, 11761074), the Natural Science Foundation of Jilin Province of China (Grant No.2020122336JC), Project of Jilin Science and Technology Development for Leading Talent of Science and Technology Innovation in Middle and Young and Team Project (No. 20200301053RQ) and the Project through the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institution.

http://doi.org/10.3837/tiis.2021.02.001 **ISSN : 1976-7277**

1. Introduction

The emergence of cloud computing and 5G communication technology have caused the transformation of the automotive industry. The Internet of Vehicles (IoV) [1] devices have further computational and service capability. Specifically, they can provide wireless communication services for vehicle terminals, roadside units (RSUs), and pedestrians in intelligent transportation systems, realizing vehicle to vehicle (V2V), vehicle to infrastructure (V2I), vehicle to person (V2P), and vehicle to network (V2N) communication modes [2].

Nevertheless, the drawbacks of cloud computing are gradually exposed with the explosive growth of vehicles and vehicle equipment. The traditional cloud computing solutions [3, 4] with high latency and low-reliability are difficult to meet users' higher requirements for transmission bandwidth and data processing delay in terms of real-time and security. It is easy to cause frequent traffic jams and traffic accidents. Also, due to the limited computing capability of automobile terminal, the increasing vehicle data has become an essential factor in restricting the development of intelligent transportation.

To cope with this disturbing problem, the emerging technique, i.e., mobile edge computing (MEC) [5-7], is applied to support delay-critical services and compute-intensive applications. The fusion of MEC and IoV technologies has developed into vehicular edge computing (VEC). Driven by VEC technology [8, 9], computing resources have been pushed to the edge of RSUs. The task generated during vehicle driving can be executed locally or offloaded to MEC, making up for the shortcomings of considerable transmission delay and unstable connection of traditional centralized IoV network [10, 11].

Computing offloading of VEC is a current research hotspot [12-16]. However, there are some deficiencies hidden in the existing works. First of all, existing models did not consider the priority of processing when multiple tasks are concurrent, i.e., the priority of driverless cars is definitely higher than that of human-crewed vehicles. This is because driverless cars have stricter requirements for data processing delay and safe driving. Furthermore, current research ignored the limitation of MEC server resources and MEC server clusters' load balancing [17, 18].

To address the above problems and further reduce the task offloading delay to guarantee driving safety and traffic efficiency. This paper proposes a context-aware vehicular task offloading (CAVTO) optimization scheme to solve computation and communication resources allocation problem in a delay-sensitive VEC system. The main contributions in this paper are summarized as follows.

- A context-aware task offloading framework based on software defined network (SDN) and network function virtualization (NFV) technology are modeled in a collaborative VEC system. Furthermore, the optimization function of delay minimization is formulated as an NP-hard problem.
- A differential evolution (DE) is proposed to solve the joint optimization problem of offloading decision and resource allocation. The ARIMA model is used to predict idle computing resources to cooperate vehicular task offloading, improve resource utilization, and reduce system delay.
- The performance of the context-aware vehicular task offloading (CAVTO) optimization scheme is evaluated by comparing it with other baseline algorithms. Simulation results show that the CAVTO scheme could generate an allocation strategy close to the optimal.

This paper is organized as follows: An overview of the related work is summarized in Section 2. Section 3 presents the offloading framework and formulates the system model.

Problem solution and algorithm proposal in Section 4. Experiment and simulation in Section 5. At last, a summary of this paper is presented in Section 6.

2. Related Works

The computing offloading of VEC is a research hotspot. The current computing offloading strategies can be divided into the following aspects: delay minimization, energy consumption minimization, weighing energy consumption and delay. According to different models, design one or multiple optimization goals and then use appropriate algorithms to solve them.

From the perspective of offloading architecture design. Sun et al. [19] considered offloading tasks to service vehicles with the same moving direction within a specific communication range. An ALTO algorithm based on MAB theory was proposed to achieve offloading delay minimization. Research [20, 21] employed vehicle cloud and remote cloud collaborative offloading method, the task is offloaded to vehicle cloud preferentially, and then offloaded to the remote cloud when computing resources are insufficient, [21] consider the heterogeneity of vehicle based on [20]. Zhu et al. [22] deployed base stations in different regions, vehicles entering and leaving must be reported to the base stations, and the base stations are responsible for task scheduling. Huang et al. [23] creatively proposed a PVEC algorithm that utilized parked vehicles as idle edge computing nodes and formulated a resource scheduling optimization problem.

From the perspective of offloading strategy optimization goal. Sun et al. [24] formulated a mixed-integer nonlinear programming problem to maximize the system's offloading utility with the joint optimization of offloading decision and task scheduling. Zhang et al. [25] proposed an offloading framework by considering the heterogeneous and the mobility of vehicles. The optimization goal is to minimize the total cost of vehicle task offloading under the constraints delay. Dai et al. [26] divided the offloading problem into two sub-problems: specifically, the optimal selection of VEC servers and joint optimization of load balancing and offloading decisions. Tang et al. [27] minimized the system delay under the condition of energy constraints. A decision tree algorithm is proposed to solve the task deployment sub-problem, and a dynamic programming technique is proposed to solve the delayed offloading sub-problem. Guo et al. [28] proposed a novel resource allocation mechanism for edge computing resource providers, which performs task offloading under the condition of observing the resource constraints on the edge server, takes the supplier's income as the optimization target, and then formulates the resource allocation problem.

From the perspective of the offloading algorithm. Liu et al. [29] used a semi-Markov decision process and linear programming to solve the optimal multi-resource allocation problem. Klaimi et al. [30] proposed a dynamic resource allocation algorithm based on game theory, which minimized CPU resource and energy consumption from the aspects of delay and request blocking probability. Finally, the existence of Nash equilibrium was proved. Feng et al. [31] adopted an ant colony optimization to solve the task scheduling problem in the AVE framework. Tham et al. [32] designed a load balancing optimization method based on a convex optimization algorithm, which improved the convergence speed and optimized the average system utilization. Wei et al. [33] combined Q-learning with DNN to optimize the problem of computing offloading in wireless cellular networks. Li et al. [34] adopted the ADMM method to study how to perform regression analysis when training samples are kept secret on the source device.

In the above research scheme, the impact of idle resources on the overall computing offloading performance has been ignored. In addition, the edge servers' load balance is not

considered on the premise of guaranteeing user service quality to improve the system operation efficiency from the perspective of the service provider. Finally, the communication resources, computing resources, and storage capacity of the vehicular edge computing network are limited, and if ignore the reasonable allocation of such resources, it will not be able to deal with the enormous data generated by the vehicle equipment.

3. System Architecture and Problem Formulation

In this section, a context-aware task offloading framework based on SDN and NFV technology is modeled in the collaborative VEC system. Furthermore, the delay minimization problem is formulated by optimizing MEC server resource allocation and vehicle task offloading decisions.

3.1 System Architecture

As illustrated in **Fig. 1,** a vehicular task offloading framework with multiple SCs and multiple RSUs has been considered in this paper. The MEC servers are deployed within the range of RSUs to provide computing services for resource-constrained vehicles. The RSUs can communicate with multiple vehicles simultaneously through massive multiple-input multiple-output (MIMO) [35, 36] technology. The virtual machine is deployed on each MEC server to facilitate centralized management of the network environment to realize real-time data perception and rapid response.

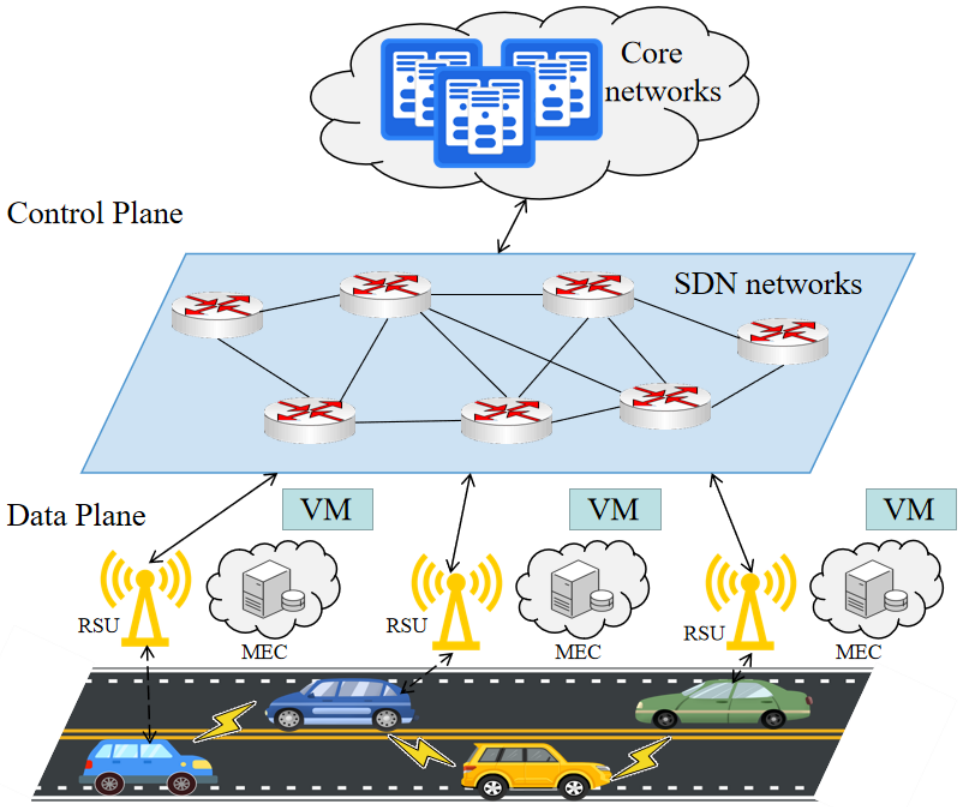

Fig. 1. System model of CAVTO architecture

In order to further improve system resource utilization, SDN and NFV technology are used to support VEC system architecture. SDN is a new type of network design concept [37] that uses a layered idea to separate the data plane from the control plane, realizes openness and programmability, and breaks the closure of traditional network equipment. NFV [38] builds many types of network equipment (such as servers, switches, storage, etc.) into a data center network and forms a virtual machine (VM) [39] by borrowing IT virtualization technology to make it run on standard server virtualization software so that it can be installed anywhere in the network without the need to deploy new hardware devices.

Overall, the SDN/NFV-based architecture can be divided into three parts. The bottom is Data Plane, which comprises date collection from vehicle ad-hoc network (VANET), RSUs, and MEC servers. The middle layer is Control Plane, and the SDN controller uniformly manages the vehicle and road information collected by RSUs. The top layer is the core network layer, supporting information sharing and service migration scheduling between MEC servers.

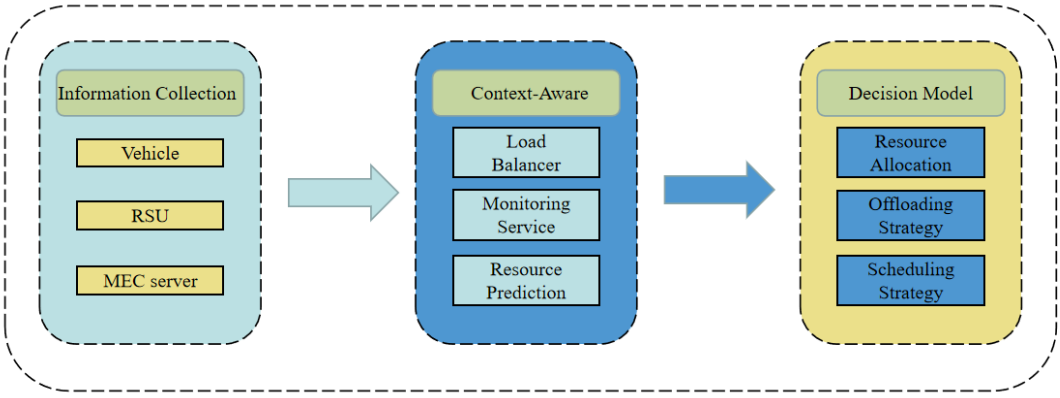

Fig. 2. Context-aware task offloading decision model

The **Fig. 2** shows the detailed function modules in the control plane. It is made up of three main components, namely, Information Collection, Context-Aware, and Decision Model.

• *Information Collection*: this module is mainly responsible for the extraction and collection of information, includes the speed, direction, and driving attributes of the vehicle. It also perceives information such as the accessible RSU within the vehicle communication range, as well as the computing resources and communication resources of the attached MEC services.

• *Context-Aware*: as the core part of the entire computing offloading framework, this module provides load balancing, real-time monitoring, resource awareness, and prediction services. The load balancer can improve server performance and effectively prevent data packet loss, processing speed limit or even collapse caused by server overload. The monitoring service provides real-time monitoring of devices' operation in the edge network. In addition, the ARIMA algorithm is employed in this module to learn the MEC servers' load changes and predict idle computing resources to achieve resource scheduling.

• *Decision Model*: according to the results of context-aware, make corresponding resource allocation strategy, offloading strategy and scheduling strategy, and send them to RSUs and vehicles for execution.

3.2 Offloading Model

Table 1. Model parameters

Parameters	Definition
$Task_{ij}$	The j-th computation task of i-th SC
X_{ij}	X_{ij}=1 if the k-th task of j-th SC is offloaded to MEC server. Otherwise, X_{ij}=0
b	The data size of the computing task
w	The priority of task processing
t^{limit}	The delay constraint
r_m	The coverage of RSU
l_m	The vertical distance from RSU to the roadside
f_m	The computing capability of the MEC server.
T_{ij}^{local}	Local processing tasks delay
T_{ij}^{stay}	The time from entry to departure within the coverage of RSU
r^{up}	The uplink transmission rate
T_{ij}^{up}	The uplink transmission delay
T_{ij}^{com}	The computing delay of task processing
r^{down}	The downlink transmission rate
T_{ij}^{down}	The downlink transmission delay

Table 1 shows some important parameters in the vehicular edge computing model, we assume that there are *n* SCs driving on the road. Each SC has *k* tasks to handle, the *j*-th computation task of *i*-th SC is denoted by $Task_{ij}$,where $i \in N$, $N = \{1,2,\cdots,n\}$ and $j \in K$, $K = \{1,2,\cdots,k\}$. For each $Task_{ij}$, it can be represented by a tuple $\{b, w, t^{limit}\}$, where *b* represents the data size of the computing task, *w* represents the priority of task processing, the purpose is to distinguish this task as a traditional computing task or a safety-oriented computing task, t^{limit} is the delay constraint. There are also *m* RSUs evenly distributed on the side of the road. Each RSU equipped with a MEC server can be regarded as a service node. The heterogeneity of RSU and MEC servers enables service nodes to have different coverage and computing capabilities. For each service nodes S_m , it can be represented by a tuple $\{r_m, l_m, f_m\}$, in which r_m is the coverage of RSU, l_m is the vertical distance from RSU to the roadside, and f_m is the computing capability of the MEC server.

Due to the limited computing resources of the vehicle itself, it is not enough to support the completion of the entire computing task locally. Offloading tasks to nearby RSUs with MEC servers is an effective solution. For each independent subtask $Task_{ij}$, its offloading strategy can be expressed as:

$$X_{ij} = \{0,1\} \tag{1}$$

where $X_{ij} = 0$ denotes the task will be executed locally, and $X_{ij} = 1$ denotes the task will be offloaded to RSUs for processing.

The entire offloading process is divided into three stages. First, the SCs upload the

computation task to the RSUs through the V2I communication method. Then, RSUs transfer the task to the MEC servers and take advantage of powerful computing resources to process the task. Finally, the computation result will be returned to SCs. Therefore, the delay in the entire offloading process is mainly caused by the task upload delay, the computing delay on MEC servers, and the result return delay. Since RSUs and MEC servers are connected by optical fiber, the transmission delay is negligible.

3.2.1 Local Computing Model

When the subtask $Task_{ij}$ of the vehicle is executed locally, the local computing delay can be expressed as :

$$T_{ij}^{local} = \frac{(1 - X_{ij}) \cdot b_{ij} \cdot C^{local}}{f_i^{local}} \tag{2}$$

where b_{ij} is the date size of $Task_{ij}$, C^{local} means CPU cycles per bit of vehicle, and f_i^{local} is the computing capacity of *i*-th SC.

3.2.2 MEC Computing Model

According to the scene shown in **Fig. 3**, the vehicle C_i travels at a constant speed at the speed of V_i. Due to the mobility of the vehicle, the distance between the vehicle and the center of RSU is constantly changing. The T_i^{stay} is the time from entry to the departure of the vehicle C_i within the coverage of RSU, which can be specifically expressed as:

$$T_i^{stay} = \frac{2\sqrt{r^2 - l^2}}{v_i} \tag{3}$$

When the vehicle reaches the coverage area of the RSUs, the SCs communicate with RSUs through LTE-V2I mode. According to the Shannon formula, the uplink transmission rate can be calculated as:

$$r^{up} = B_{ij} \cdot \log_2(1 + \frac{p_i \cdot h_i}{B_{ij} \cdot N_\sigma}) \tag{4}$$

where B_{ij} denotes allocated channel bandwidth, P_i is the transmit power of i-th SC, and h_i means channel gain.

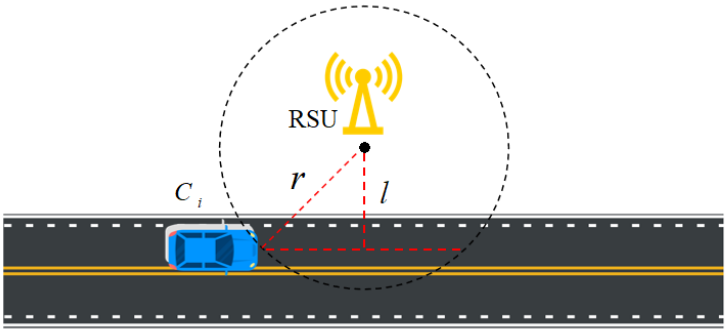

Fig. 3. Driving map of the vehicle within the coverage of RSU

The transmission delay of the uplink can be expressed as:

$$T_{ij}^{up} = \frac{d_{ij} \cdot X_{ij}}{r^{up}} \tag{5}$$

After the vehicular task is uploaded to the RSU, computing resources are provided through the attached MEC server. Therefore, the computing delay of task processing can be expressed as:

$$T_{ij}^{com} = \frac{d_{ij} \cdot X_{ij} \cdot C^{mec}}{f_m^{mec}} \tag{6}$$

where C^{mec} implies the number of CPU cycles of unit data in the MEC system and f_m^{mec} represents computing resource allocated by *m*-th MEC server.

Finally, after the task has been processed, the MEC servers return the computing result to SCs. The downlink transmission delay is:

$$T_{ij}^{down} = \frac{\alpha \cdot d_{ij} \cdot X_{ij}}{r^{down}} \tag{7}$$

where α is the ratio of the size of the upload task and the returned computing result.

It is worth noting that the MEC servers can only start processing the task after the server has received the task of WDs completely, and the MEC servers can only start sending back the computing results at the end of completing the entire computing task.

In summary, the total computing offloading delay of processing $Task_{ij}$ can be expressed as:

$$T_{ij}^{mec} = T_{ij}^{up} + T_{ij}^{com} + T_{ij}^{down} \tag{8}$$

3.3 Problem Formulation

Based on the above-mentioned offloading model, the goal of this article is to minimize the average task processing delay of the vehicles through the joint optimization of computing resource allocation and transmission bandwidth allocation. An objective function *Obj* is

introduced, which can be expressed as:

$$Obj = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k}(T_{ij}^{local} + T_{ij}^{mec})}{nk} \tag{9}$$

The final computing offloading problem model can be established as:

$$\begin{aligned} &\min\ Obj(X_{ij}, B_{ij}, f_m^{mec}, w_i) \\ &\text{s.t.}\quad C_1: X_{ij} = \{0,1\} \\ &\qquad C_2: 0 \le \sum_{i=1}^{n} B_{ij} \le B_{total} \\ &\qquad C_3: 0 \le \sum_{m=1}^{m} f_m^{mec} \le f_{total} \\ &\qquad C_4: T_i^{mec} \le T_i^{stay} \\ &\qquad C_5: \max \sum_{i=1}^{n} w_i \end{aligned} \tag{10}$$

where constraint C_1 is the offloading decision of vehicle, constraint C_2 means the allocated bandwidth cannot exceed the total bandwidth, constraint C_3 means the allocated computing resource cannot exceed the total resource of MEC servers, constraint C_4 is set to ensure that the task processing will not be interrupted, the computing task is required to be completed before the vehicle leaves the range of the RSU, constraint C_5 is to maximize the weight, that is, to deal with the higher priority security tasks preferentially.

4. Proposed Scheme

In this section, we propose a CAVTO optimization scheme to solve the above optimization problem. The CAVTO optimization scheme is mainly divided into two parts. Firstly, a differential evolution algorithm (DE) is proposed to optimize offloading decision and resource allocation in collaborative vehicular edge computing networks to minimize the average delay. Secondly, the ARIMA-based machine learning algorithms are used to predict idle computing resources to ensure MEC server load balance and improve utilization efficiency. The specific expression is as follows.

4.1 Latency Optimization

The optimization problem of the above equation (10) is an NP-hard problem. We consider adopting the DE algorithm to get the optimal solution. DE algorithm is a heuristic search algorithm based on the biological evolution process. Simulate the problem to be solved as a biological evolution process, and find the optimal solution through evolution. DE algorithms usually start with a set of possible potential solutions, which are composed of genetically encoded individuals. After fitness calculation, selection, crossover, and mutation, these individuals evolve from generation to generation to produce better approximate solutions. **Algorithm 1** shows detailed steps.

Algorithm 1 Latency Optimization Based on Differential Evolution Algorithm

Input: Population size: N, Dimension: M, Max iteration: G, Number of SCs: n, Number of tasks: k, Total bandwidth: B_{total}, Total computing resources: f_M.

Output: The optimal solution: $S_{ij}^{*}=\{X_{ij}, B_{ij}, f_m^{mec}\}$

The minimal delay: Obj^{*}

$g=0$
for $i=1$ to N **do**
  for $j=1$ to M **do**
    Initialize population $S_{ij}(0)$
  end
end
while $(g \le G)$ **do**
  for $i=1$ to N **do**
    for $j=1$ to M **do**
      Mutation according to Eq. (14);
      Crossover according to Eq. (15);
    end
    Selection according to Eq. (13);
  end
  $g=g+1$
End
return S_{ij}^{*}, Obj^{*}

The floating-point encoding is employed to replace traditional binary encoding, which can effectively reduce storage space and reduce algorithm complexity. The chromosome coding method is shown in **Fig. 4,** the total number of computing tasks for all vehicles is set to the chromosome length, each gene represents a computing task $Task_{ij}$, and the value of the corresponding gene S_{ij} represents offloading strategy X_{ij}, bandwidth allocation strategy B_{ij} computing resource allocation strategy f_m^{mec} and task processing priority w_i, whose strategy set can be expressed as:

$$S_{ij}=\{X_{ij}, B_{ij}, f_m^{mec}\} \tag{11}$$

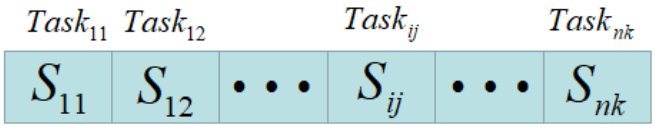

Fig. 4. Chromosome encoding

After the population coding is completed, the fitness needs to be set. The greater the fitness, the more chance it will be inherited to the next generation. Since the objective function is to minimize the average delay of vehicular offloading, the fitness is set as:

$$Fit = \frac{1}{Obj} \tag{12}$$

The principle of selection is that the higher fitness, the more likely to be selected. According to the roulette wheel selection, the selection probability is expressed as:

$$P(i) = \frac{Fit(i)}{\sum_{i=1}^{N} Fit(i)} \tag{13}$$

The mutation operation is to change the gene position, so that the algorithm has the ability of local random search and avoid premature convergence. Randomly select three different individuals $S_{r1}(g), S_{r2}(g), S_{r3}(g)$ in the strategy set, the produced intermediate expressed as:

$$I_i(g+1) = S_{r1}(g) + F \cdot (S_{r2}(g) - S_{r3}(g)) \tag{14}$$

where F is the scaling factor.

The purpose of crossover operation is to increase the diversity of solutions, The result of crossover between $S_i(g)$ and $I_i(g+1)$ is:

$$H_i(g+1) = \begin{cases} I_i(g+1) & rand(0,1) \le CR \\ S_i(g) & otherwise \end{cases} \tag{15}$$

where CR is crossover probability.

Through the above steps, the solution set of the strategy $S_{ij} = \{X_{ij}, B_{ij}, f_m^{mec}\}$ evolves from generation to generation, producing better and better approximate solutions. Finally, record the optimal solution S^*, and calculate the minimum average delay Obj^*.

4.2 Resource Prediction

In context-aware collaborative vehicular edge computing networks, the SDN controller can monitor MEC servers load changes in real time. In different regions, the traffic flow at different time periods is very different, which leads to server load unbalanced. In order to improve resource utilization, adaptively learning the load changes of the server and realizing resource prediction and scheduling is an effective way to improve traffic conditions. Therefore, the ARIMA model is applied to predict the number of vehicles arriving in the area in the next time period based on the historical data of traffic flow.

The basic idea of the ARIMA model [40] is that the time-varying data sequence is a random sequence, which can be described by a certain mathematical model to predict the future value from the past value and the current value of the time series. The ARIMA (p,d,q) model can be expressed as:

$$y_T = \mu + \sum_{i=1}^{p} Y_i y_{T-i} + \varepsilon_T + \sum_{i=1}^{q} \theta_i \varepsilon_{T-i} \tag{16}$$

where p is the autoregressive order, d denotes the difference times and q is the average moving order.

Suppose there are R regions in the vehicular edge network, which can be represented by set $\{1,2,\cdots r,\cdots R\}$. $P^T(A_r)$ is the traffic flow of region A_r in period T. According to the traffic flow over the past T periods $P^1(A_r), P^2(A_r), \cdots, P^T(A_r)$, the traffic flow in period $T+1$ can be predicted. The greater the traffic flow in a period of time, the higher the server load in the area. After the prediction is completed, the sequence $P^{T+1}(A_1), P^{T+1}(A_2), \cdots, P^{T+1}(A_r)$ can be get. In descending order according to traffic flow, take out the top H regions and bottom L regions. Dispatch the computing resources in the L regions to the H regions, and cooperate to complete the computing offloading.

The resource prediction based on ARIMA model mainly consists of 4 steps.

Step 1: Check whether the data sequence over the past T periods $P^1(A_r), P^2(A_r), \cdots, P^T(A_r)$ is stationary. If the series is nonstationary, perform d order difference operation until a stationary sequence is obtained.

Step 2: Determine ARIMA model parameters p and q according to sequence characteristics.

Step 3: Calculate the autocorrelation function and partial correlation function of the time series, check whether the ARIMA model (p,d,q) is satisfied.

Step 4: If satisfied, forecast the traffic flow $P^{T+1}(A_r)$ in period $T+1$. Otherwise, repeat Step 2 and Step 3.

Step 5: Sort sequence $P^{T+1}(A_1), P^{T+1}(A_2), \cdots, P^{T+1}(A_r)$ in descending order to implement resource scheduling.

4.3 CAVTO Optimization Scheme

Combining the DE-based delay optimization algorithm and ARIMA-based resource prediction algorithm, the CATVO optimization scheme is proposed in **Algorithm 2**. First, based on historical data, the traffic flow of different areas in the next time period can be predicted. The traffic flow indirectly reflects the load condition of the server in this region. Then, low-load MEC servers assists in performing computing offloading for high-load MEC servers. Finally, the DE algorithm is adopted to jointly optimize resource allocation and offloading decisions. The result shows that the CATVO optimization scheme not only effectively reduces the offloading delay, but also improves resource utilization.

Algorithm 2 CAVTO optimization scheme

Input: Population size: N, Dimension: M, Max iteration: G, Number of SCs: n, Number of tasks: k, Total bandwidth: B_{total}, Total computing resources: f_M. Data of traffic flow $P^1(A_r), P^2(A_r), \cdots, P^T(A_r)$

Output: The optimal solution: $S_{ij}^*(T+1) = \{X_{ij}, B_{ij}, f_m^{mec}\}$

The minimal delay: $Obj^*(T+1)$

1 Predict $P^{T+1}(A_1), P^{T+1}(A_2), \cdots, P^{T+1}(A_r)$ according to ARIMA algorithm

2 Sort sequence $P^{T+1}(A_1), P^{T+1}(A_2), \cdots, P^{T+1}(A_r)$ in descending order
3 Scheduling the computing resources from L regions to the H regions
4 Adopt algorithm.1
5 **Return** $S_{ij}^{*}(T+1)$ and $Obj^{*}(T+1)$

5. Simulation Results and Analysis

In this section, we set the environmental parameters of the simulation experiment and evaluate the performance of the CAVTO optimization scheme by comparing other baseline algorithms. The specific details are shown as follows.

5.1 Parameters Settings

The scenario assumed in this article is on a one-way traffic road. The task of the driving vehicle can be executed locally or offloaded to the RSUs. Due to the heterogeneity of vehicles, assume that the number of computing tasks for the vehicle is 5-10. The computing capability of each vehicle is randomly distributed as $4 \times 10^6 \sim 2 \times 10^7$ cycles/s, and each car travels at a certain speed. The specific simulation parameters are shown in **Table 2**.

Table 2. Simulation parameters

Parameters	Value
RSU coverage radius	100~200/m
Vertical distance from RSU to road	30~50/m
Transmission bandwidth	5 MHz
Vehicle computing capability	$4 \times 10^6 \sim 2 \times 10^7$ cycles/s
Vehicle transmission power	1.3 W
Gaussian white noise power	3×10^{-13} W
Channel gain	4
Number of tasks unit vehicle	5~10
MEC server computing capability	$8 \times 10^7 \sim 2 \times 10^8$ cycles/s
Task weight coefficient	0~3
Data size of vehicular task	100 kb
Task computing capability	50 cycles/bit

5.2 Experiment Results

Suppose there are five vehicles driving RSU in the same time period. In order to allocate resources reasonably, make wise offloading decisions. The DE algorithm is employed to minimize the average system delay. As shown in **Fig. 5**, we initialize the population size to 100, evolution algebra is set as 150, and reorganization probability is 0.1. The experimental results show that the optimal objective function value gets the minimum 1806 ms after 140 iterations. The detailed allocation strategy of the DE algorithm is shown in **Fig. 6**.

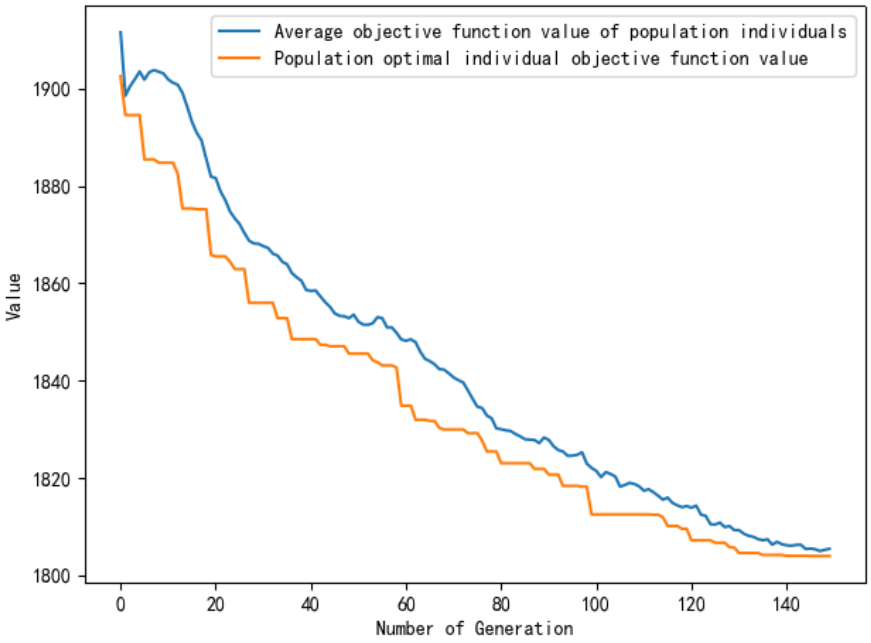

Fig. 5. The DE offloading scheme

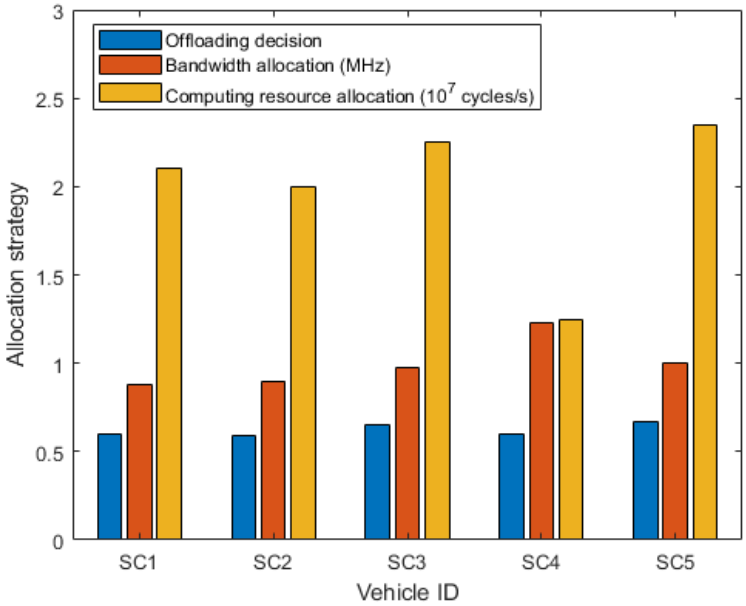

Fig. 6. Allocation strategy of DE algorithm

In terms of idle resource prediction, we use the ARIMA model to predict the traffic flow in the next period. The data set is provided by the Korea Expressway Corporation [41], the data format is shown in **Fig. 7**. This data records the hourly traffic flow on a certain highway.

Traffic Section	Time	2020.04.22	2020.04.21	2020.04.20	2020.04.19	2020.04.18	2020.04.17	2020.04.16	2020.04.15
Seoul TG→ Shingal JC	00	1,452	1,398	1,339	1,461	1,806	1,488	1,287	1,832
	01	1,027	960	906	983	1,290	1,148	901	1,363
	02	724	746	662	662	937	823	676	1,000
	03	716	722	745	588	843	739	736	845
	04	1,039	1,110	1,682	689	1,206	943	1,422	1,046
	05	3,166	3,193	4,768	2,081	3,403	2,448	4,230	2,762
	06	5,476	5,531	6,264	3,195	5,113	4,799	6,343	4,399
	07	5,470	5,423	5,163	2,960	4,991	5,431	6,045	4,198
	08	5,622	5,419	5,003	2,973	5,407	5,029	5,540	3,959
	09	5,449	5,350	4,725	3,852	5,262	4,864	5,108	4,681
	10	6,166	5,733	5,975	4,810	5,843	4,961	5,761	5,658
	11	6,157	5,193	5,481	4,885	4,935	5,300	5,302	5,423
	12	5,201	5,555	4,915	4,571	5,297	4,945	5,133	4,821
	13	4,593	5,185	4,512	4,786	5,138	4,915	4,998	4,571
	14	5,096	5,095	4,298	5,200	5,066	4,204	5,082	4,561
	15	4,986	5,081	4,351	4,669	5,214	4,575	5,019	4,521
	16	5,010	4,848	4,613	4,005	5,346	4,003	5,145	4,128
	17	5,373	4,993	5,079	3,684	4,949	4,348	5,277	4,062
	18	5,255	5,474	5,458	3,834	4,612	4,679	5,532	3,754
	19	4,950	5,027	5,040	3,318	3,833	4,486	5,061	3,393
	20	4,173	4,519	4,449	3,556	3,660	4,257	3,975	3,550
	21	3,571	3,554	3,514	4,284	3,568	4,029	3,668	3,377
	22	2,962	2,997	2,824	3,439	3,205	3,202	2,998	2,881
	23	2,121	2,134	2,038	2,158	2,162	2,364	2,232	1,913

Fig. 7. Traffic flow data set

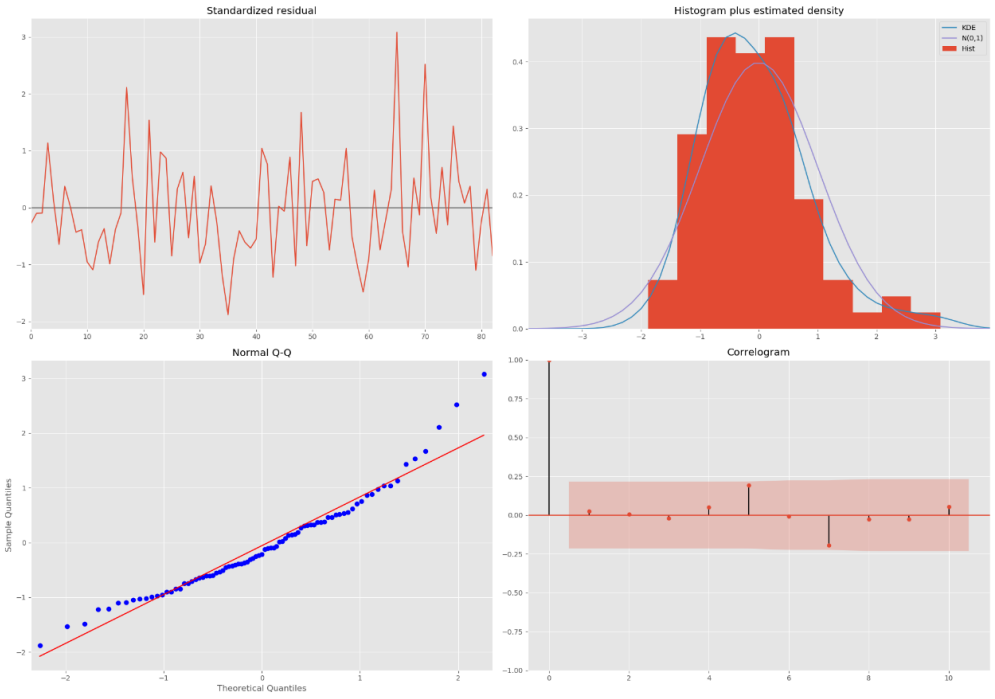

Fig. 8. Model sequence diagram

After initial processing of the data, we calculate the characteristic of standardized residual, histogram plus estimated density, autocorrelation function, and partial correlation function, etc. The purpose is to analyze the reliability and periodicity of the data, and check whether the data obey the normal distribution. The result is as shown in **Fig. 8**. The calculation shows that the data set has good stationarity and is suitable for the ARIMA model.

Fig. 9 presents the traffic flow prediction based on the ARIMA model. According to the traffic flow in the past six days, predict the traffic flow situation within 24 hours. Set the time node with less traffic as the idle resource node of this region, and dispatch computing resources to other regions to improve offloading efficiency.

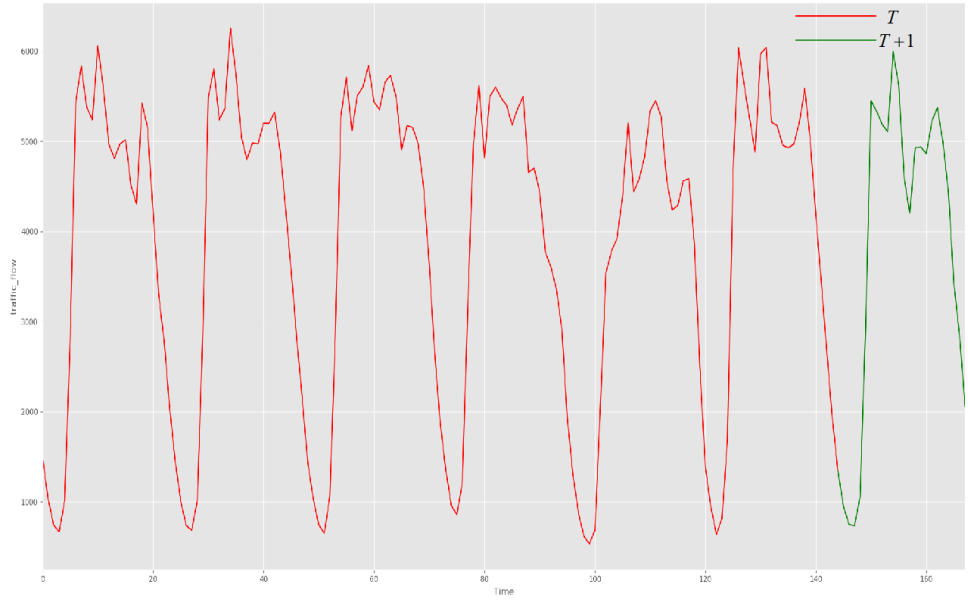

Fig. 9. Traffic flow prediction based on ARIMA model

5.3 Algorithm Performance Evaluation

In order to further evaluate the performance of the CAVTO optimization scheme, compare CAVTO with the following offloading schemes: 1) Local Execution (LE): all the computing tasks of vehicle execute locally. 2) MEC Execution (ME): all the computing tasks will be offloaded to RSUs for execution. 3) Differential Evolution algorithm (DE): which without considering resource forecast and schedule.

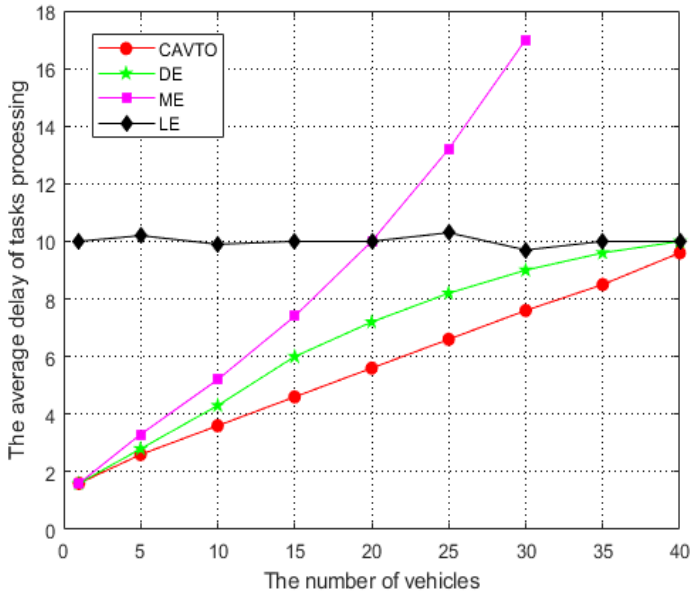

Fig. 10. The average delay of tasks processing with the increasing of vehicles

The **Fig. 10** reveals the relationship between the number of vehicles and the average delay. It can be seen from the diagram, as the number of vehicles increases, the CAVTO scheme and DE algorithm show a slow upward trend; the DE algorithm is inferior to CATVO in average delay because there is no prediction and scheduling of idle resources. The ME strategy offloads all computing tasks to RSUs. In the beginning, the delay optimization performed well. However, as the number of vehicles increased, channel congestion was caused, and the supply of resources exceeded demand, which causes a large delay. In summary, CAVTO is superior to the other three algorithms and reduces the average delay by up to 16% compared with the DE strategy.

Furthermore, in order to study the number of vehicular tasks with different priorities, the total weight ($\sum_{i=1}^{n} w_i$) of task completion under different offloading schemes has been researched through repeated experiments.

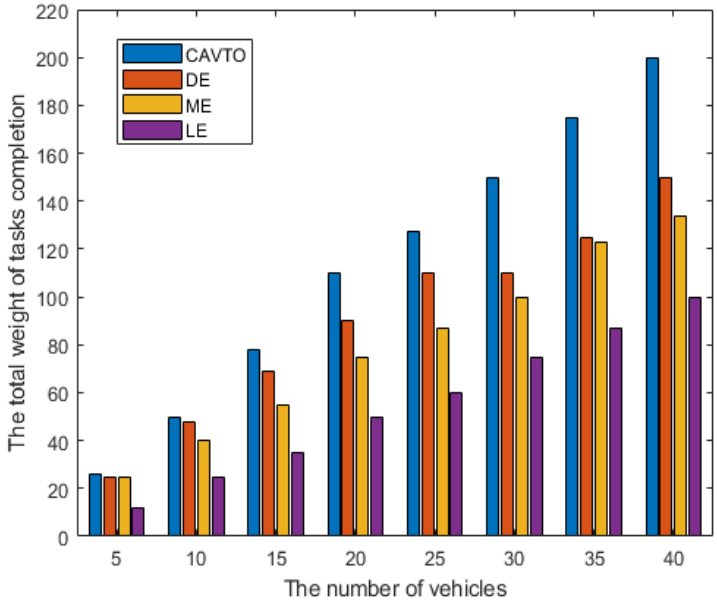

Fig. 11. The total weight of tasks completion under different offloading schemes

It can be seen from **Fig. 11**, the total weight of the CAVTO optimization scheme is the largest, which explains that the strategy pays more attention to priority processing of high-priority tasks. Compared with DE, ME, LE strategies that do not consider priority, the CAVTO optimization scheme has increased by 20%, 31%, and 52%, respectively.

6. Conclusion

In this paper, an effective CAVTO optimization scheme is proposed in a vehicular edge computing system with multiple SCs and multiple MEC servers. Technologically, an improved differential evolution algorithm is designed to figure out the joint optimization problem of offloading decision and resource allocation. Furthermore, the ARIMA-based machine learning algorithms are used to predict idle computing resources to ensure MEC server load balance and improve utilization efficiency. The experimental results demonstrate that the CAVTO optimization scheme could generate a near-best resource allocation strategy by comparing baseline algorithms and reduce the system delay significantly.

References

[1] J. Cheng, 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, Oct. 2015. Article (CrossRef Link)

[2] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "LTE for vehicular networking: a survey," *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148-157, May 2013. Article (CrossRef Link)

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, Feb. 2018. Article (CrossRef Link)

[4] H. Zhang, G. Chen, and X. Li, "Resource management in cloud computing with optimal pricing policies," *Computer Systems Science and Engineering*, vol. 34, no. 4, pp. 249-254, 2019. Article (CrossRef Link)

[5] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54-61, Apr. 2017. Article (CrossRef Link)

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016. Article (CrossRef Link)

[7] 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. Article (CrossRef Link)

[8] J. Zhang and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246-261, Feb. 2020. Article (CrossRef Link)

[9] W. Tong, A. Hussain, W. X. Bo, and S. Maharjan, "Artificial Intelligence for Vehicle-to-Everything: A Survey," *IEEE Access*, vol. 7, pp. 10823-10843, Jan. 2019. Article (CrossRef Link)

[10] H. Gao, W. Huang, and X. Yang, "Applying Probabilistic Model Checking to Path Planning in an Intelligent Transportation System Using Mobility Trajectories and Their Statistical Data," *Intelligent Automation and Soft Computing*, vol. 25, no. 3, pp. 547-559, 2019. Article (CrossRef Link)

[11] J. Liu, X. Kang, C. Dong, and F. Zhang, "Simulation of Real-Time Path Planning for Large-Scale Transportation Network Using Parallel Computation," *Intelligent Automation and Soft Computing*, vol. 25, no. 1, pp. 65-77, 2019. Article (CrossRef Link)

[12] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep-Reinforcement-Learning-Based Offloading Scheduling for Vehicular Edge Computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449-5465, June 2020. Article (CrossRef Link)

[13] X. Cao, H. Yu, and H. Sun, "Dynamic Task Assignment for Multi-AUV Cooperative Hunting," *Intelligent Automation and Soft Computing*, vol. 25, no. 1, pp. 25-34, 2019. Article (CrossRef Link)

[14] Y. Jang, J. Na, S. Jeong, and J. Kang, "Energy-Efficient Task Offloading for Vehicular Edge Computing: Joint Optimization of Offloading and Bit Allocation," in *Proc. of the 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1-5, May 2020. Article (CrossRef Link)

[15] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint Optimization of Computation Offloading and Task Scheduling in Vehicular Edge Computing Networks," *IEEE Access*, vol. 8, pp. 10466-10477, Jan. 2020. Article (CrossRef Link)

[16] S. Choo, J. Kim, and S. Pack, "Optimal Task Offloading and Resource Allocation in Software-Defined Vehicular Edge Computing," in *Proc. of International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 251-256, Oct. 2018. Article (CrossRef Link)

[17] S. Zaman, T. Maqsood, M. Ali, K. Bilal, S. Madani, and A. Khan, "A load balanced task scheduling heuristic for large-scale computing systems," *Computer Systems Science and Engineering*, vol. 34, no. 2, pp. 79-90, 2019. Article (CrossRef Link)

[18] M. Okhovvat and M. Kangavari, "TSLBS: A time-sensitive and load balanced scheduling approach to wireless sensor actor networks," *Computer Systems Science and Engineering*, vol. 34, no. 1, pp. 13-21, 2019. Article (CrossRef Link)

[19] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive Learning-Based Task Offloading for Vehicular Edge Computing Systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061-3074, Apr. 2019. Article (CrossRef Link)
[20] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An SMDP-Based Resource Allocation in Vehicular Cloud Computing Systems," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7920-7928, Dec. 2015. Article (CrossRef Link)
[21] C. Lin, D. Deng, and C. Yao, "Resource Allocation in Vehicular Cloud Computing Systems with Heterogeneous Vehicles and Roadside Units," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3692-3700, Oct. 2018. Article (CrossRef Link)
[22] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4150-4161, June 2019. Article (CrossRef Link)
[23] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked Vehicle Edge Computing: Exploiting Opportunistic Resources for Distributed Mobile Applications," *IEEE Access*, vol. 6, pp. 66649-66663, Nov. 2018. Article (CrossRef Link)
[24] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint Optimization of Computation Offloading and Task Scheduling in Vehicular Edge Computing Networks," *IEEE Access*, vol. 8, pp. 10466-10477, Nov. 2020. Article (CrossRef Link)
[25] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36-44, June 2017. Article (CrossRef Link)
[26] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint Load Balancing and Offloading in Vehicular Edge Computing and Networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377-4387, June 2019. Article (CrossRef Link)
[27] D. Tang, X. Zhang, and X. Tao, "Delay-Optimal Temporal-Spatial Computation Offloading Schemes for Vehicular Edge Computing Systems," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1-6, Oct. 2019. Article (CrossRef Link)
[28] Y. Guo, F. Liu, N. Xiao, and Z. Chen, "Task-Based Resource Allocation Bid in Edge Computing Micro Datacenter," *Computers, Materials and Continua*, vol. 61, no. 2, pp. 777-792, 2019. Article (CrossRef Link)
[29] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398-2410, Oct. 2016. Article (CrossRef Link)
[30] J. Klaimi, S. Senouci, and M. Messous, "Theoretical Game Approach for Mobile Users Resource Management in a Vehicular Fog Computing Environment," in *Proc. of the 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 452-457, Aug. 2018. Article (CrossRef Link)
[31] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous Vehicular Edge Computing Framework with ACO-Based Scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660-10675, Dec. 2017. Article (CrossRef Link)
[32] C. Tham and R. Chattopadhyay, "A load balancing scheme for sensing and analytics on a mobile edge computing network," in *Proc. of IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1-9, July 2017. Article (CrossRef Link)
[33] Y. Wei, Z. Wang, D. Guo, and F. R. Yu, "Deep Q-Learning Based Computation Offloading Strategy for Mobile Edge Computing," *Computers, Materials and Continua*, vol. 59, no. 1, pp. 89-104, 2019. Article (CrossRef Link)
[34] Y. Li, X. Wang, W. Fang, F. Xue, H. Jin, Y. Zhang, and X. Li, "A Distributed ADMM Approach for Collaborative Regression Learning in Edge Computing," *Computers, Materials and Continua*, vol. 59, no. 2, pp. 493-508, 2019. Article (CrossRef Link)
[35] L. N. Ribeiro, S. Schwarz, M. Rupp, and A. L. F. de Almeida, "Energy Efficiency of mmWave Massive MIMO Precoding with Low-Resolution DACs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 2, pp. 298-312, May 2018. Article (CrossRef Link)

[36] S. Schwarz, M. Rupp, and S. Wesemann, "Grassmannian Product Codebooks for Limited Feedback Massive MIMO With Two-Tier Precoding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 5, pp. 1119-1135, Sep. 2019. Article (CrossRef Link)
[37] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN Networks: A Survey of Existing Approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259-3306, May 2018. Article (CrossRef Link)
[38] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A Survey on Emerging SDN and NFV Security Mechanisms for IoT Systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 812-837, Aug. 2019. Article (CrossRef Link)
[39] T. Ma, S. Pang, W. Zhang, and S. Hao, "Virtual Machine Based on Genetic Algorithm Used in Time and Power Oriented Cloud Computing Task Scheduling," *Intelligent Automation and Soft Computing*, vol. 25, no. 3, pp. 605-613, 2019. Article (CrossRef Link)
[40] P. Shu and Q. Du, "Group Behavior-Based Collaborative Caching for Mobile Edge Computing," in *Proc. of IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chongqing, China, pp. 2441-2447, May 2020. Article (CrossRef Link)
[41] Korea Expressway Corporation. [Online]. Available: http://data.ex.co.kr/portal/traffic/trafficVds#

Zilong Jin received the B.E. degree in computer engineering from the Harbin University of Science and Technology, China, in 2009, and the M.S. and Ph.D. degrees in computer engineering from Kyung Hee University, South Korea, in 2011 and 2016, respectively. He is currently an Assistant Professor with the School of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include wireless sensor networks, mobile wireless networks, and cognitive radio networks.

Chengbo Zhang received his B.E. degree in computer science from Nanjing University of Information Science and Technology, China, in 2019. Now he is a master student in School of Computer and Software, Nanjing University of Information Science & Technology. His main research interests include internet of mobile communication and edge computing.

Guanzhe Zhao received B.E. degree in the Department of Electrical Engineering from Yan Shan University, China, in 2009, and M.S. degree in the Department of Electronics and Radio Engineering from Kyung Hee University, South Korea, in 2011. He is currently a Lecturer in Huihua College of Hebei Normal University. His research interests are in communication systems, cognitive radio networks, and deep learning.

Yuanfeng Jin received the B.S. and the M.S. degree in Department of Mathematics from the YanBian University, China, in 1998 and 2004, respectively, and Ph.D. degrees in Department of Applied Mathematics from Kyung Hee University, South Korea, in 2010. He is currently a professor in Mathematics at Departmet of Mathematics of Yanbian University, China. His research interests are numerical algorithms, partial differential equations and scientific computation.

Lejun Zhang received the M.S. degree in computer science and technology from the Harbin Institute of Technology and the Ph.D. degree in computer science and technology from Harbin Engineering University. He is currently a Professor with Yangzhou University, China. His research interests include computer networks, social network analysis, and information security.