Emotion-aware Task Scheduling for Autonomous Vehicles in Software-defined Edge Networks

Mengmeng Sun[1,2], Lianming Zhang[1], Jing Mei[1*], and Pingping Dong[1*]
[1]College of Information Science and Engineering, Hunan Normal University
Changsha, 410081, China
[2]College of Cultural Communication, Henan Vocational Institute of Arts
Zhengzhou, 451464, China
[e-mail: 1311631214@qq.com, zlm@hunnu.edu.cn, JingMei1988@163.com ppdong@hunnu.edu.cn]
[*]Corresponding author: J. Mei and P. Dong

Received April 20, 2022; revised August 18, 2022; revised September 22, 2022; revised October 21, 2022; accepted November 5, 2022; published November 30, 2022

Abstract

Autonomous vehicles are gradually being regarded as the mainstream trend of future development of the automobile industry. Autonomous driving networks generate many intensive and delay-sensitive computing tasks. The storage space, computing power, and battery capacity of autonomous vehicle terminals cannot meet the resource requirements of the tasks. In this paper, we focus on the task scheduling problem of autonomous driving in software-defined edge networks. By analyzing the intensive and delay-sensitive computing tasks of autonomous vehicles, we propose an emotion model that is related to task urgency and changes with execution time and propose an optimal base station (BS) task scheduling (OBSTS) algorithm. Task sentiment is an important factor that changes with the length of time that computing tasks with different urgency levels remain in the queue. The algorithm uses task sentiment as a performance indicator to measure task scheduling. Experimental results show that the OBSTS algorithm can more effectively meet the intensive and delay-sensitive requirements of vehicle terminals for network resources and improve user service experience.

Keywords: software-defined edge network, autonomous vehicles, emotion model, task scheduling

This research is supported in part by the grant from the National Natural Science Foundation of China (Nos. 61572191 and 61602171), and Hunan Provincial Natural Science Foundation of China (Nos. 2022JJ30398, 2022JJ40278 and 2022JJ40277).

http://doi.org/10.3837/tiis.2022.11.004 **ISSN : 1976-7277**

1. Introduction

Autonomous driving is a technology for cooperatively manipulating cars in road environment perception, location calculation, and autonomous decision-making. Since the 1970s, autonomous driving technology has been investigated, and recent developments have been particularly remarkable. Google started developing self-driving vehicles in 2009 and demonstrated a fully functional, autonomous driving prototype car in 2014 [1]. One of the most promising advances in automotive engineering and research—the development of self-driving cars—has spread across universities and the automotive industry worldwide over the past decade [2]. With the development of autonomous vehicles, the increase in the number of autonomous driving terminals will generate massive amounts of data. According to Huawei's forecast, approximately 180 ZB of new data will be generated worldwide every year by 2025 [3]. The application of autonomous driving generates higher performance requirements for computing and storage resources in networks.

Cloud computing has problems such as high energy consumption and poor real-time performance due to the high round-trip time caused by the long-distance communication link. In contrast, edge computing is a three-layer computing model based on the cloud layer, edge layer, and terminal layer [4]. Edge computing does not need to transmit tasks to the cloud data center via long-distance communication links. Edge servers deployed near smart terminals and equipped with faster 5G wireless networks can exchange data within milliseconds and can provide low-latency and high-efficiency services to nearby terminal devices [5]. With these high-quality network characteristics, edge computing technology has good application prospects in autonomous driving [6].

The task carried out by the autonomous vehicle has strict response time requirements due to its urgency and external environmental influence factors. However, due to factors such as numerous tasks, insufficient computing power, and limited battery capacity in autonomous vehicles, it is difficult to support intensive and delay-sensitive computing tasks. Therefore, it is a key issue to migrate these tasks to an edge server near the smart terminal for computing so that the edge server can execute and complete these delay-sensitive tasks for autonomous driving as much as possible. The simple scheduling algorithm migrates tasks to the nearest edge server using the principle of proximity. However, the computing capacity of the edge server is limited [7]. If numerous autonomous driving tasks are concentrated, the edge server may be overloaded. To avoid vehicle safety hazards, an important method to achieve load balancing is task scheduling [8]. The separation of the control plane and data plane of software-defined networking (SDN) realizes the flexible management of complex networks [9]. The SDN can be effectively integrated with the edge computing system architecture with the advantages of global control and coordination to more flexibly manage network resources. In view of the problems between the execution ability and resource requirements of autonomous vehicle computing tasks and research, it is urgent to carry out task scheduling. In this paper, we integrate the respective advantages of SDN and edge computing and investigate scheduling strategies for computing tasks of autonomous vehicles in software-defined edge networks. The main contributions of this paper are listed as follows:

- We propose a framework of software-defined edge autonomous driving networks. This framework integrates the advantages of SDN and edge computing. With the separation technology of the control plane and the data plane of the network equipment, the network resources are centrally managed, and the task scheduling decision is more effectively reached.

- We introduce the concept of emotion and establish an emotion model for scheduling tasks of autonomous vehicles as one of the innovations of this paper. By analyzing different categories of computing tasks for autonomous vehicles, an emotion model related to task urgency is established. The emotion changes with the length of time that computing tasks with different urgency levels remain in the queue. As an important factor for constraining task scheduling, emotion satisfies the delay requirements of autonomous vehicles for computing tasks.
- We develop an optimal base station (BS) task scheduling (OBSTS) algorithm for a software-defined edge autonomous driving network. In the OBSTS algorithm, the SDN controller allocates the optimal edge server resources for its prescheduling, compares the emotion generated by the task queue waiting time with the emotion of the task itself, and selects the best BS that meets the task scheduling.

The remainder of this paper is organized as follows: We introduce the related work in Section 2. Section 3 describes the system model and its problem statement. In Section 4, we develop the OBSTS algorithm. Extensive experiments are conducted in Section 5. Section 6 provides the conclusion and future work.

2. Related Works

With the rapid development of the Internet of Things, many researchers have performed much research on the architecture of edge computing. All references in related works are analyzed in **Table 1**.

Table 1. Research analysis of related works

Number	Research content	Pros and cons
[10]	The optimization goal is to minimize the total waiting time.	In-depth research on optimization problems can add important influencing factors.
[11]	A multistage greedy adjustment algorithm is proposed.	
[12]	An efficient task scheduling algorithm is developed.	Paper [12-13] conducts good research on task scheduling but can also optimize on latency and energy consumption for user result return.
[13]	A stochastic optimization problem constrained by queue stability and energy is formulated.	
[14-16]	The factors affecting the total delay in the task scheduling process are analyzed.	The impact of task result return delay on total delay is improved, and various constraints such as queue backlog and deadline during task scheduling can still be examined.
[17]	An architecture based on device-to-device cooperative mobility is proposed.	
[18]	A scheme is proposed to maximize the task to meet the delay requirement.	
[19]	Multiserver mobile edge computing Internet of Vehicles is researched.	These papers have been optimized for the waiting time of the task backlog, and reasonable selection of task computing resources on
[20]	A hybrid dynamic scheduling scheme is proposed.	

[21]	A lightweight heuristic solution is proposed.	edge servers can be considered.
[22]	A random load scheduling framework is developed.	
[23]	A task placement strategy and scheduling algorithm are proposed.	
[24]	Edge computing strategies using SDN and network functions virtualization are investigated.	The attributes of tasks and real-time scheduling is comprehensively considered to minimize the total system delay and energy consumption.
[25-26]	SDN and edge computing frameworks are integrated into vehicle networks to improve vehicle service latency.	
[27]	A reputation incentive mechanism based on software-defined vehicular edge computing is proposed.	How edge servers filter from many services is an important research point.
[28]	A fast search algorithm based on genetic algorithm is proposed.	The limitation is that each vehicle can only belong to one volunteer union.
[29]	An inviting algorithm for contributors is proposed to recruit cooperators through social closeness.	
[30]	A wirelessly powered mobile edge computing system is proposed.	Paper [30] focuses on minimizing total energy consumption in wireless power supply.

Kao et al. [10] proposed a complete polynomial-time approximation scheme to solve the tradeoff between delay sensitivity and energy cost of mobile devices. The scheme uses a directed graph to represent multiple tasks and minimizes the total waiting time as the optimization goal. SahniLin et al. [11] investigated the assignment problem of joint scheduling of data-aware tasks and network cooperative flow edge computing and proposed a multistage greedy adjustment (MSGA) algorithm. The joint problem is mathematically modeled in this paper to minimize the overall completion time of the application. When addressing multitask scheduling problems, they are formulated as an NP-hard optimization problem. To solve the optimization problem, Liu et al. [12] developed an efficient task scheduling algorithm. The basic idea is to prioritize tasks to ensure the completion time constraints of applications and the processing dependencies of tasks and to reduce the average completion time of multiple applications. Chen et al. [13] established a stochastic optimization problem constrained by queue stability and energy in a two-layer edge computing system composed of a macro BS and multiple micro BSs and focused on joint optimization of task scheduling and energy management decision-making. However, their approach disregards the delay and energy consumption of the result returning to the user after the task scheduling is completed.

To improve the impact of the task result return delay on the total delay in the task scheduling process, the total weighted response time of task scheduling, including the delay caused by task placement, scheduling processing, task result return, and the delay caused by I/O interruption, were comprehensively considered [14-16]. To fully consider end user mobility, Saleem et al. [17] proposed an edge computing architecture based on the device-to-device cooperation mobile, taking into account the user's mobility, distributed resources, task attributes, user equipment energy consumption, and other constraints. The architecture uses

mobile sensing adjacent idle resources to obtain low-complexity effective task scheduling, reducing the task calculation delay. However, it will be a challenge for multiple users of different applications to compete for shared resources. To solve the task scheduling and resource allocation problems of different users, Zhao et al. [18] separately analyzed the scheduling problem of delay-sensitive tasks in single-user and multiuser scenarios and proposed an optimization scheme that maximizes the probability of tasks meeting the delay requirements. The resource coordination between the edge cloud and the remote cloud improves the success rate of user tasks, but the task scheduling process does not comprehensively consider various constraints, such as queue backlogs and deadlines.

The waiting time of the task backlog in the server queue is also an important consideration. Deng et al. [19] explored the multiserver mobile edge computing Internet of Vehicles, which solves the task scheduling optimization problem of minimizing the task completion time with the specified cost. Chen et al. [20] proposed a hybrid dynamic scheduling scheme, including a queue-based dynamic scheduling algorithm and time-based dynamic scheduling, to select the server with the fastest response. To solve the problem of task execution delay on the mobile edge network side, Wang et al. [21] used task scheduling to reduce the execution delay of tasks in mobile edge networks and proposed a lightweight heuristic solution. In [22], considering the inherent trade-off between communication and computational load, a stochastic load scheduling framework was developed, and an enhanced Lagrangian method was utilized to obtain the optimal computational load scheduling algorithm. Li et al. [23] proposed task placement strategies and scheduling algorithms to reduce task calculation delay and response time.

The application scenarios of autonomous vehicles also need to comprehensively consider the special attributes of the task and the mobility of the vehicle. For the task scheduling problem, it is also necessary to consider the reasonable selection of the task computing resources on the edge server. To be suitable for the management of massive data access networks, the development of data processing has had a substantial impact on the demand and evolution of infrastructure networks, which has enabled the expansion of a new paradigm of edge computing. Lv et al. [24] elaborated an edge computing strategy from the perspective of SDN and network function virtualization. The authors in [25-26] integrated a SDN and edge computing framework into a vehicle network, focusing on improving the delay of vehicle service. Zeng et al. [27] proposed a new SDN-based vehicle edge computing framework, which introduced reputation to measure the contribution of each vehicle as a basis for providing different quality of service. Papers [28-29] mainly study interaction modeling between two contributors optimized for two Stackelberg games. Mao et al. [30] proposed a wirelessly powered mobile edge computing system.

Through research and analysis of the existing work, we identified deficiencies in comprehensive consideration factors in the task solutions generated by autonomous vehicles, such as centralized management of resources, delay in returning task results, and urgency of tasks. To better solve the task scheduling problem generated by autonomous vehicles, we comprehensively consider the attributes of the tasks and real-time scheduling in this paper so that the reasonable execution and processing of computing tasks is the key issue in handling an autonomous vehicle. The software-defined edge computing architecture is selected to complete the centralized scheduling of resources, the tasks are classified according to the urgency of the tasks, and the optimal BS task scheduling algorithm is designed to complete the processing of autonomous vehicle tasks.

3. System Description

3.1 System Model

The SDN architecture is an established paradigm. For example, related definitions are presented in [31-33]. However, in this paper, we integrate the SDN and edge computing to solve the task scheduling problem in autonomous driving. In the following section, we introduce the software-defined paradigm into edge computing and depict a framework of software-defined edge networks for autonomous driving, as shown in **Fig. 1**. The framework includes the cloud layer, terminal layer, and edge layer where the SDN controller is deployed. The edge server and cloud center are connected via the backbone network, and the vehicle and the edge server will communicate with the wireless link. We embed the edge computing network in the SDN control layer and use the idea of separating the SDN control plane from the data plane to logically realize the centralized management function of the edge computing network. The framework not only achieves global information of edge network resources and optimize resource allocation strategies but also demonstrates the characteristics of low latency, high bandwidth, low energy consumption, and safety and reliability. The computing service of edge networks sinks to the terminal, and the edge server and terminal data are transmitted via the wireless network. The area where the edge server provides services for the terminal can be described as a circular range with the geographic location, where the server is deployed as the center and the wireless signal coverage length as the radius. Vehicles can hand over intensive computing tasks, require high real-time performance, and exceed their processing capabilities to the edge server for collaborative execution. Therefore, how to choose a suitable edge server to assist the terminal in processing computing tasks is very important. The SDN controller has the topology information of the edge network and the running state of the vehicles and can flexibly control and schedule the resources of edge networks. The SDN controller and vehicles are connected through the north-bound interface to provide services for applications, and the SDN controller and edge infrastructure are connected through the southbound interface to realize the update of global status information [34]. During the operation of the autonomous vehicle, the SDN controller can allocate the optimal edge server for the terminal tasks that need to be dispatched, process the tasks, and realize the smooth and safe operation of the autonomous vehicle.

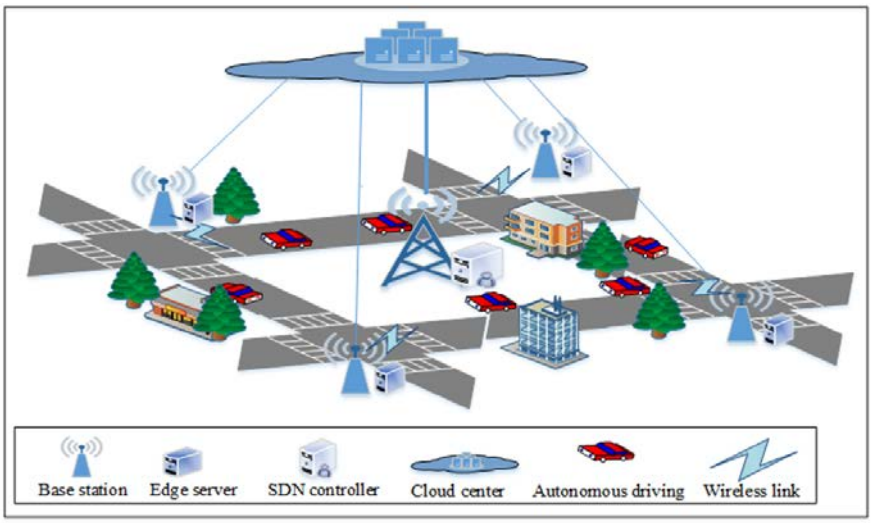

Fig. 1. Framework of software-defined edge networks for autonomous vehicles.

In this paper, the emotion of the computing task of autonomous vehicles is used to characterize the performance of the queuing of computing tasks. The longer the task remains in the queue waiting for processing, the greater the emotion of the task. There is a certain correlation between the emotion and the urgency of the task. When the urgency of the task is higher, the execution constraint time of the task is shorter, and the emotion of the task is greater.

Different autonomous vehicle tasks have different types of computing tasks, constraint times, and levels of urgency and emotion. The classification of computing tasks for autonomous vehicles is shown in **Table 2**. Computing tasks can be divided into three types: level-I tasks, level-II tasks, and level-III tasks. Level-I tasks belong to the category of real-time and extremely high security, such as collision protection, emergency braking, and blind-spot detection. The emotion of level-I tasks is assigned a maximum. The urgency of noncontrol tasks, such as navigation, call, music, and video, is relatively low. These tasks are at level II or level III.

Table 2. Classification Table of calculation tasks for autonomous vehicles

Type of task	Importance	Urgency	Emotionality
Collision protection, emergency braking, blind spot detection, road selection	Very important	I	high
Navigation, call, voice, information	important	II	middle
Music, video, broadcasting, entertainment	Generally, important	III	low

The emotion of a task is related to the two basic attributes of its task category and queue waiting time. The definition of task emotion is shown in Equation (12). The change in the emotion of the three-level tasks with the queuing time is shown in **Fig. 2**. As the waiting time of the tasks in the three types increases, the emotion of the tasks show an exponential increase, and the tasks tend to stabilize after being backlogged in the queue for a certain period. The slope k of the curve can indicate how quickly the task emotion changes with the queue time. The slope k_1 of the level-I curve is the largest, the slope k_2 of the level-II curve is the second largest, and the slope k_3 of the level-III curve is the smallest. Taking into account the safety of autonomous vehicles and the quality of experience, there is a certain upper bound for the waiting times of tasks in queue times t_1, t_2, and t_3. At this time, the emotions corresponding to the three types of tasks are d_1, d_2, and d_3.

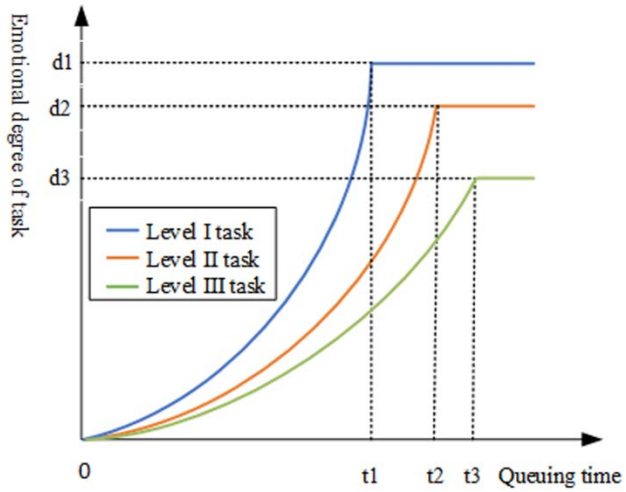

Fig. 2. Emotion vs. queuing time.

3.2 Problem description

To model the task scheduling for autonomous driving in the software-defined edge network framework, the computing tasks are defined as

$$Job = \{J_{size}, J_{cpu}, J_{req}, J_{arr}, J_{loc}\} \tag{1}$$

where J_{size} is the size of the task, J_{cpu} is the number of CPU cycles required by the task, J_{req} is the request time of the task, J_{arr} is the arrival time of the task, and J_{loc} is the spatial position when the task is generated, which is represented by coordinates (X, Y). For convenience, the main symbols in this paper are described in **Table 3**.

Table 3. Description of main symbols.

Notation	Description
S	Edge server set
V	Autonomous driving vehicle set
T	Task set
SNR	SNR of transmission link
P_v^k	Vehicle transmission power
$h_{v,s}^k$	Channel gain
σ^2	Gaussian white noise power
$R_{v,s}^k$	Link transmission rate
B	Channel bandwidth
$T_{tra_{up}}$	Time delay of task upload to BS
$E_{tra_{up}}$	Task upload energy consumption of BS
T_e	Task execution time
T_w	Queue waiting time of the task

T_{pro}	Task completion processing time
ϑ_{urg}	Task urgency
w	Energy consumption of vehicles
$D_{emo}^{s_j}$	Emotion value of the nearest server
D_{emo}^{J}	Emotion value of the task

3.2.1 Transmission Rate between Vehicles and BSs

In the software-defined edge autonomous driving network, high-speed communication between vehicles and BSs is carried out via the 5G network. Each vehicle generates several computing tasks and is scheduled to the nearest or neighboring BS. Therefore, the task scheduling decision of autonomous vehicles is defined as follows:

$$A = \{a_{v,s} | v \in V, s \in S\} \tag{2}$$

where $a_{v,s} \in \{0,1\}$, v belongs to the set V of autonomous vehicles, and s belongs to the set S of edge servers. When $a_{v,s} = 0$, the computing task on vehicle v is scheduled to the server of the nearest BS. When $a_{v,s} = 1$, the computing task on vehicle v is scheduled to the server of the neighboring BS. Each edge server simultaneously provides services for multiple vehicles, but a vehicle can only select one server for task scheduling at any time, and we obtain

$$\sum_{s \in S} a_{v,s} \leq 1, \forall v \in V \tag{3}$$

Assuming that each edge server has k subchannels, the set of subchannels can be expressed as $K = \{1, 2..., k\}$, the bandwidth of each subchannel is B, and the signal-noise ratio (SNR) of the vehicle's data transmission on subchannel k is expressed by

$$SNR_{v,s}^{k} = \frac{p_v^k h_{v,s}^k}{\sigma^2} \tag{4}$$

where p_v^k represents the data transmission power of vehicle v on channel k, $h_{v,s}^k$ represents the channel gain and σ^2 represents the Gaussian white noise power.

According to Shannon's theorem [35], the computing task generated by vehicle v is scheduled to server s of the BS, and when channel k is selected for data transmission, the transmission rate $R_{v,s}^k$ of the link is

$$R_{v,s}^{k} = Blog_2(1 + SNR_{v,s}^{k}) \tag{5}$$

3.2.2. Delay and Energy Consumption of Upload Tasks to BS

When the vehicle generates a computing task and needs to send a request to the edge layer, its uplink transmission delay is expressed as follows:

$$T_{tra_{up}} = \frac{J_{size}}{R_{v,s}^k \frac{R}{q_{v,s}}} \tag{6}$$

where R represents the service range of the BS and $q_{v,s}$ represents the link distance between vehicle v and server s. The autonomous vehicle needs to consume some energy when transmitting tasks. The transmission energy consumption $E_{tra_{up}}$ is the product of the transmission power and the transmission time, which is expressed as follows:

$$E_{tra_{up}} = P_i^{t_0} T_{tra_{up}} \tag{7}$$

where $P_i^{t_0}$ is the transmission power of the automatic driving vehicle transmission task at the moment.

3.2.3. Delay and Energy Consumption for Processing Tasks

The time for processing tasks is expressed as the sum of the execution time of the task and the waiting time of the task. The execution time T_e of the task is the ratio of the calculation amount J_{size} of the task to the calculation capacity c_s of the edge server. We obtain

$$T_e = \frac{J_{size}}{c_s} \tag{8}$$

The waiting time T_w of the task is the sum of the processing time from the current task to the previous task. We have

$$T_w = \sum_{j=1}^{n} \frac{J_{size}}{c_s} \tag{9}$$

where $n = j$ -1, and $\sum_{j=1}^{n} \frac{J_{size}}{c_s}$ is the sum of the processing time of the previous j -1 tasks. The time T_{pro} for processing tasks is defined as

$$T_{pro} = T_e + T_w \tag{10}$$

The energy consumption Epro for completing tasks is expressed as

$$E_{pro} = E_u J_{size} T_{pro} \tag{11}$$

where E_u is the energy consumption of edge server nodes for computing tasks in unit time.

3.2.4. Emotion of Scheduling Tasks

During the period from the start of the task scheduling to the completion of the task scheduling, as the waiting time in the queue increases and the urgency of the task itself is generated, the task sentiment will produce an anxiety emotional state, which is positively related to the urgency of the task. The related relationship-ship is expressed in Equation (12).

$$D_{emo}(t) = \vartheta_{urg}^{J} \int_0^{T_w} \xi(t) dt \tag{12}$$

where ϑ_{urg}^{J} is the urgency of the task and $\xi(t)$ is the correlation between the urgency and the emotion of the task. $\int_0^{T_w} \xi(t) dt$ is the emotional accumulation value during the period when the task is waiting in the queue.

3.2.5. Total cost of the System

The total cost of the system is the linear weighted value of the cost of the total time and energy consumption for scheduling tasks. The total time T_{total} of task scheduling is the sum of the time $T_{tra_{up}}$ spent on task uploading to the BS and the time T_{pro} spent on task scheduling. This value will mainly include the following three parts: the transmission delay on the uplink, the processing delay of the task scheduling in the nearest BS when $a_{vs} = 1$, and the task scheduling in the neighboring BS when $a_{vs} = 0$. The total cost T_{total} of energy consumption for scheduling tasks is expressed as the sum of the energy consumption $E_{tra_{up}}$ of the task uploading BS and the energy consumption T_{pro} of the task execution. From the above findings, we have $T_{total} = T_{tra_{up}} + T_{pro}$ and $E_{total} = E_{tra_{up}} + E_{pro}$. The total cost of the system is

defined by

$$G(t) = w_1 T_{total} + (1 - w_1) E_{total} \tag{13}$$

where w_1 and $1 - w_1$ are the weight values of the total delay and the total energy consumption of the system, respectively, and $w_1 \in (0,1)$.

To optimize the performance of the system, the optimization goal of minimizing the total cost is modeled and expressed as follows:

$$\begin{aligned} min\eta &= \lim_{t\to\infty} \frac{1}{t} E\{\sum_{\tau=0}^{t-1} G(\tau)\} \\ s.t.: C_1: &\sum_{s\in S} a_{v,s} \le 1, \forall v \in V \\ C_2: &\sum E_{tra_{up}} + E_{tra_{down}} < w \end{aligned} \tag{14}$$

In Equation (14), C_1 indicates that one edge server can simultaneously serve multiple autonomous vehicles, but only one server can be selected for scheduling the same task at any time. C_2 indicates that the sum of energy consumption on the uplink and downlink is less than the vehicle's energy consumption.

4. OBSTS Algorithm

In this section, we propose an algorithm for computing task scheduling for autonomous vehicles, which is named OBSTS; its logical control structure is shown in **Fig. 3**. The autonomous vehicle prerequests the edge server to assist in computing task *J*, and the SDN controller obtains the task scheduling request. The simplest scheduling strategy is to use the nearest principle for the generated computing tasks, make the first scheduling decision, and preschedule the vehicle to the edge server S_1 associated with the BS by using the first come, first served order in the task queue. Due to the different types and delay sensitivity of computing tasks of autonomous vehicles, the emotions generated by tasks in the queue are also different. Therefore, the SDN controller needs to update the edge server set and adjust the task scheduling strategy by analyzing indicators such as the waiting delay of tasks in the edge server queue and the amount of task backlog according to the emotion of task *J*. Task *J* will make a second-step scheduling decision. The SDN controller preschedules the task at the edge server S_3 and repeats the previous step until it identifies the best BS that meets the task scheduling. Then, it uploads the task to the edge server with sufficient resources to calculate the task result to achieve the goal of efficiently processing the task and reducing the execution delay of task scheduling.

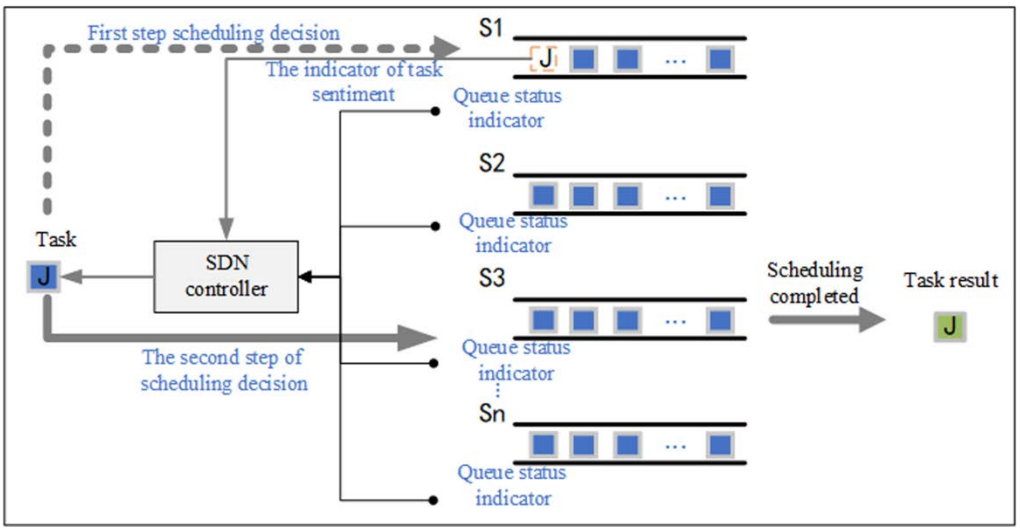

Fig. 3. Logic control for scheduling tasks on the BS.

First, we need to initialize the computing task scheduling decision, calculate the distance between each task to be scheduled and the server, select the closest server from the edge server set, and then calculate the emotional tolerance of the computing tasks in the queue. The time is compared until the edge server that meets the minimum delay and strong execution ability is identified, and the scheduling decision of the computing task is returned to complete the scheduling of the optimal BS to complete the task. The detailed process is presented as follows: When the autonomous vehicle sends a new task command, the SDN controller receives the command centralized control command. By calculating the execution time and sentiment of the new task, the SDN controller schedules the task to the BS nearest the place where the vehicle sends the command. The task enters the edge server queue and waits. The task scheduling status process is divided into the following situations: if the task processing time is less than the task sentiment, the task will continue to wait for scheduling in the BTS; if the processing time of the task is greater than or equal to the emotional level of the task, the nearest BTS cannot complete the scheduling of the new task. The SDN controller distributes the task to the adjacent BS, recalculates the queuing time of the new task, compares the processing time of the task with the queue task waiting time in the adjacent server, and determines the best BTS to complete the task scheduling when the minimum delay and strong server execution capability are met. The task scheduling completion instruction is sent to the SDN controller.

The OBSTS algorithm uses computing tasks as the research object. The execution process needs to traverse a set of computing tasks. Its time complexity reflects the efficiency of the algorithm, which is expressed as $O(n)$, where n represents the number of computing tasks to be scheduled. The pseudocode is shown in Algorithm 1.

Algorithm 1: OBSTS Algorithm

Input: Computing task set J and edge server set S
Output: Computing task scheduling decision $a_{v,s}$

1. Set $a_{v,s} = 0$
2. Calculate the distance between each task to be scheduled and the server
3. **For** J is not empty **do**
4. Select the nearest server S_j from set S
5. Calculate the calculation tasks in the queue D_{emo}^{J} and $D_{emo}^{S_j}(t)$
6. **If** $D_{emo}^{S_j}(t) > D_{emo}^{J}$ **then**
7. **For** S_j neighboring BS **do**
8. Select the nearest server from neighboring BSs S_{j+1}
9. $D_{emo}^{S_j}(t) \leftarrow D_{emo}^{S_{j+1}}(t)$
10. **If** $D_{emo}^{S_{j+1}}(t) \leq D_{emo}^{J}$ and the greatest resource demand **then**
11. S_{j+1} can be executed, and the current task is removed
12. $a_{v,s} = 1$
13. **End if**
14. **End for**
15. **Else**
16. S_j can execute, and the current task is removed
17. **End if**
18. **End for**
19. **Return** $a_{v,s}$

5. Performance Evaluation

5.1. Experimental Setup

To verify the effectiveness of our proposed algorithm, we performed many experiments. The hardware devices for these experiments include smart cars driven by Raspberry Pi 3 Model B+, laptops, and desktop hosts. The operating system running on the Raspberry Pi development board is Raspbian, and that running on desktop hosts and laptops is Ubuntu 16.04 64 bit. A laptop computer simulates edge servers and BSs, a desktop host acts as an SDN controller, and cloud computing is implemented by a high-performance desktop host. We use Java and Python language to realize the program coding of the experiment. After the experimental system is started, the SDN controller initializes the state information and maintains the topology data of the entire network. During the experiment, when the smart car terminal generates a computing task that requires the assistance of the edge BS, it sends a task scheduling request to the SDN controller. After the controller receives the request, it returns a request response message to the vehicle and simultaneously starts the scheduling algorithm to allocate the best BS for the computing task. In the experimental topology, latitude and longitude are used to identify points in the entire area, and the geographic area abstracted by latitude and longitude data is applied as the location range for vehicle operation and equipment deployment. The experimental topology is shown in **Fig. 4**.

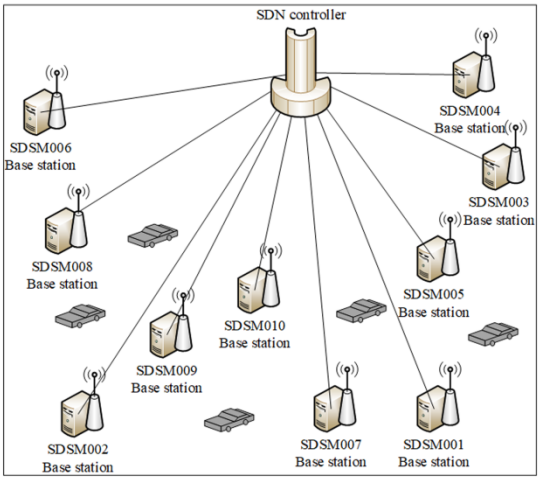

Fig. 4. Experimental topology.

The network consists of one SDN controller and ten edge BSs. The edge BSs and SDN controller are connected in a wired manner. If the BS signals have overlapping areas, the adjacent BSs can wirelessly communicate. The experimental parameters are shown in **Table 4**.

Table 4. Experimental parameter settings.

Parameter	Default value
Number of BSs	10
Number of edge servers	10
Number of SDN controllers	1
Task calculation amount (Kb)	1-1000
Edge node CPU computing capacity (GHz)	[4, 6]
Edge node storage size (G)	64
CPU cycles (Number/bit)	[400, 1000]
Edge node energy consumption (J/cycle)	$[1,8]\times 10^{-10}$
Bandwidth size (MHz)	60
Gaussian white noise (dB)	-100

5.2. Results

In this section, we discuss the queuing delay, processing delay, processing energy consumption, task backlog, and other performance of the OBSTS algorithm, and the Random algorithm, Nearest algorithm, and improved min-min algorithm (TPMM) [11] are compared. The random algorithm randomly schedules computing task requests in the BS in the edge layer for processing. The nearest algorithm schedules the computing task request in the nearest BS in the edge layer for processing according to the Euclidean distance. The TPMM algorithm matches the corresponding BS for computing and processing according to the data volume of the task to be processed, the priority of the task, and the working status of the edge server.

5.2.1. Queuing delay

The queuing delay for computing tasks of vehicles varies with the number of tasks, as shown in **Fig. 5**. As the number of tasks increases, the OBSTS and TPMM, Random, and Nearest algorithms change in a wave-like manner. When the number of computing tasks is small, the edge server can allocate processors in time, and the queuing delay of computing tasks is low. The OBSTS algorithm has the smallest growth rate with the increase in the number of tasks compared with the other three algorithms, mainly because the OBSTS algorithm can better reduce the queue length in the edge server and the queuing delay of computing tasks. The average values of the queuing delay of the OBSTS, TPMM, Random, and Nearest algorithms are 4.75 ms, 13.48 ms, 30.78 ms, and 68.88 ms, respectively. Compared with the other three algorithms, the OBSTS algorithm reduces the task queuing average delay by 64.76%, 84.56%, and 93.1%, respectively.

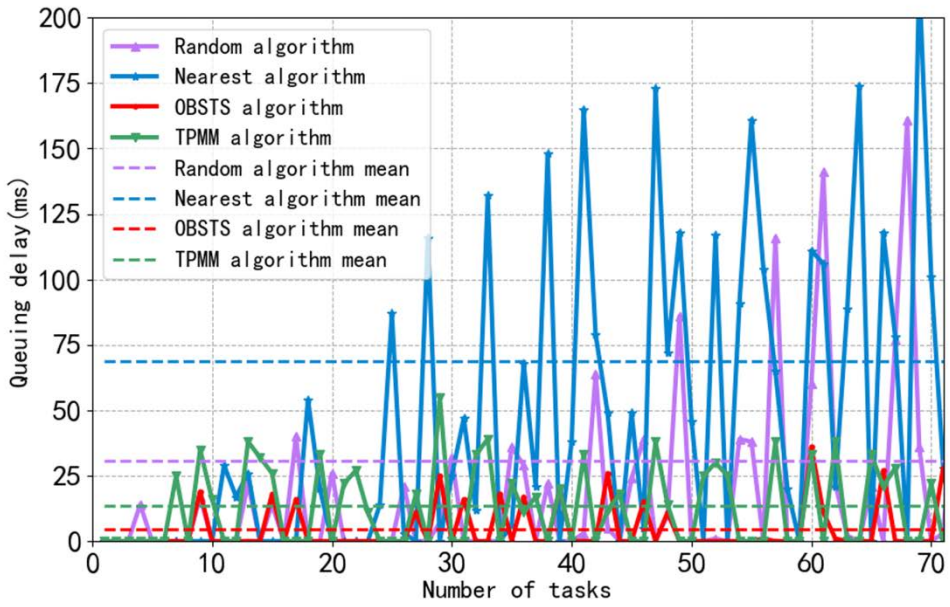

Fig. 5. Queuing delay.

5.2.2. Processing delay

The effect of the number of tasks on the processing delay is shown in **Fig. 6**. The processing delay of the OBSTS algorithm is smaller than that of the other three algorithms, and the changing trend is stable. The waiting delay of a task is mainly affected by factors such as the task length of the queue in the edge server and the scheduling strategy. The processing delay of the task is determined by the processing capacity of the CPU of the edge server. Reducing the average queue length of the queue can significantly reduce the waiting time of the computing task, thereby shortening the processing delay of the computing task and effectively completing the scheduling of the computing task. The maximum processing delays of the OBSTS, TPMM, Random, and Nearest algorithms are 38 ms, 45 ms, 91 ms, and 173 ms, respectively.

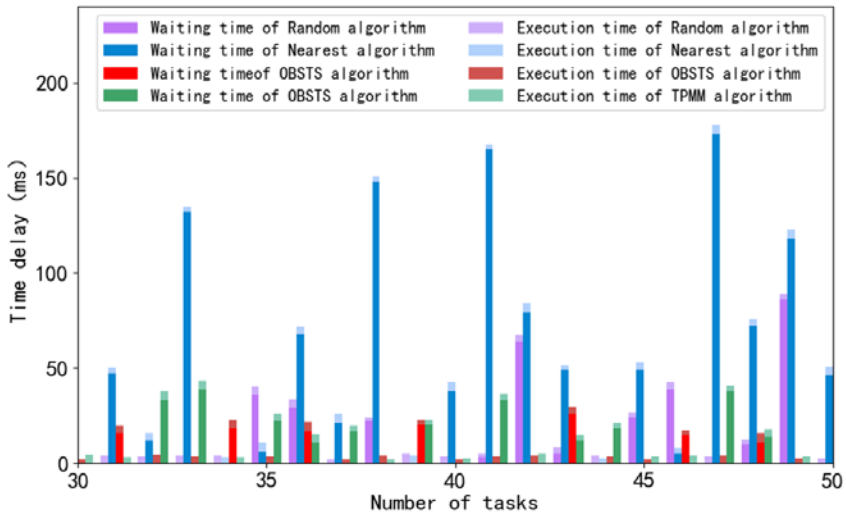

Fig. 6. Processing delay.

5.2.3. Energy consumption

The relationship between energy consumption and the number of tasks is shown in **Fig. 7**. With an increase in tasks, the processing energy consumption of the four algorithms shows an overall increasing trend, but the OBSTS algorithm has the slowest increase, and the other three algorithms have higher energy consumption, mainly due to the tasks in the edge server queues. In the task number between 29 and 43, the average values for processing energy consumption of the OBSTS, TPMM, random, and nearest algorithms are 0.0021 J, 0.0032 J, 0.0065 J, and 0.0183 J, respectively. The OBSTS algorithm reduces the processing energy consumption by 34.37%, 67.69%, and 88.52% compared with the TPMM, Random, and Nearest algorithms, respectively.

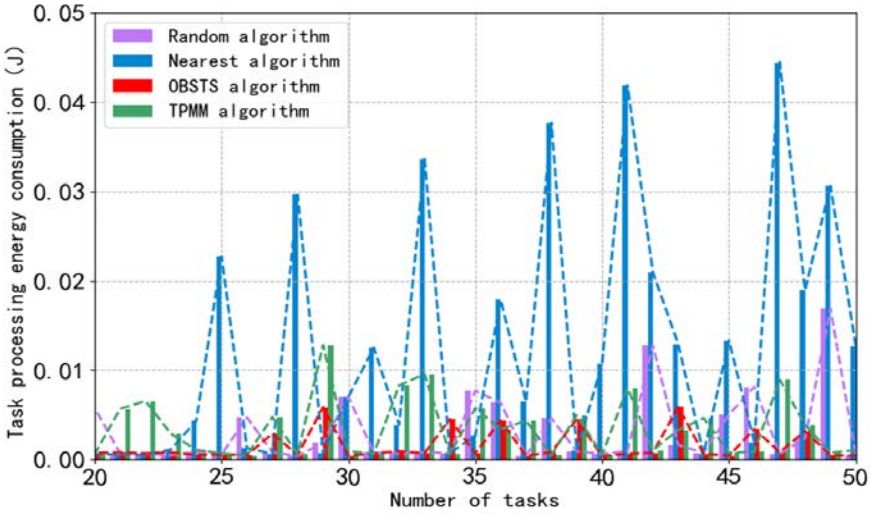

Fig. 7. Energy consumption.

5.2.4. Backlog

The queue length in the edge server reflects the backlog of computing tasks. When the task backlog is smaller, the algorithm performance is better. The change in the backlog of tasks in the server with the execution time is shown in **Fig. 8**. The four algorithms show a growth trend, but the growth rate of the task backlog of the OBSTS algorithm is the smallest. The random algorithm is affected by its random attributes. When the execution time falls between 36 ms and 50 ms, the task backlog in the queue increases at a faster rate, and the growth rate is the largest compared with the OBSTS, TPMM, and nearest algorithms. The slope of the curve is defined as the backlogging rate of tasks. The backlog rates of the OBSTS, TPMM, Random, and Nearest algorithms are approximately 1.4, 1.5, 1.8, and 2.9, respectively. In general, the OBSTS algorithm effectively alleviates the load balancing problem of edge servers.

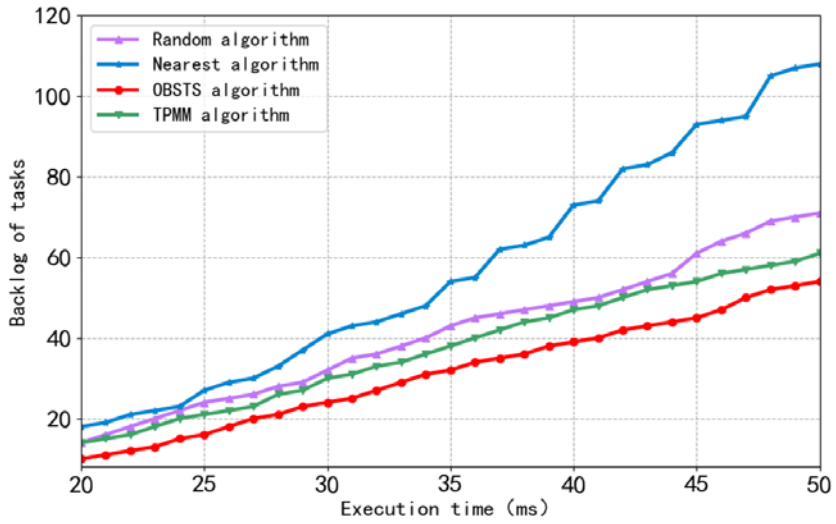

Fig. 8. Backlog.

Via the performance evaluation of the queuing delay of computing tasks, processing delay, backlog of tasks in the edge server queue, and total system delay, different algorithms are analyzed and compared, as shown in **Table 5**. On the whole, compared with the other three algorithms, the OBSTS algorithm has the lowest average queuing delay, and the task backlog rate of the queue in the edge server is only 1.4 per unit time. The OBSTS algorithm has the highest execution efficiency and the best system stability.

Table 5. Comparison of different algorithms

Influencing factors	OBSTS	TPMM	Random	Nearest
Queuing delay	4.75 ms	13.48 ms	30.78 ms	68.88 ms
Processing delay	38 ms	45 ms,	91 ms	173 ms
Energy consumption	0.0021 J	0.0032 J	0.0065 J	0.0183 J
Backlog	1.4	1.5	1.8	2.9
stability	Most stable	Relatively stable	Relatively stable	Unstable
effectiveness	High	Middle	Low	Low

6. Conclusion and Future Work

Aimed at the computing task scheduling problem in an autonomous driving network, this paper combines the advantages of a SDN and edge computing to design a framework of a software-defined edge autonomous driving network. The tasks are classified according to the importance of the computing tasks of autonomous vehicles, and the emotion model associated with the task urgency is established. The emotion of the task is regarded as an important factor to restrict the task scheduling, and the OBSTS algorithm is proposed. The experimental results show that the OBSTS algorithm achieves the minimum total system delay and energy consumption. In future research, we can further explore the cooperative interaction between autonomous vehicles and the safety and reliability of tasks so that the computing task scheduling of autonomous vehicles is more accurate and reliable.

References

[1] Q. Luo, Y. Cao, J. Liu, and A. Benslimane, "Localization and navigation in autonomous driving: Threats and countermeasures," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 38-45, 2019. Article (CrossRef Link)

[2] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826-1848, 2020. Article (CrossRef Link)

[3] Autonomous driving network solution white paper, May. 2, 2020. [Online] Available: https://www-file.huawei.com/-/media/corporate/pdf/news/autonomous-driving-network-whitepaper.pdf?la=zh.

[4] 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, 2016. Article (CrossRef Link)

[5] 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)

[6] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8. pp. 1697-1716, 2019. Article (CrossRef Link)

[7] P. Yang, N. Zhang, S. Zhang, Li Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915-929, 2019. Article (CrossRef Link)

[8] C. Tseng, F. Tseng, Y. Yang, C. Liu, and L. Chou, "Task scheduling for edge computing with agile VNFs on-demand service model toward 5G and beyond," *Wireless Communications and Mobile Computing*, vol. 2018, 2018, Article ID 7802797. Article (CrossRef Link)

[9] Y. Li, and M. Chen, "Software-defined network function virtualization: a survey," *IEEE Access*, vol. 3, pp. 2542-2553, 2015. Article (CrossRef Link)

[10] Y. Kao, B. Krishnamachari, M. Ra, and F. Bai, "Hermes: latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056-3069, 2017. Article (CrossRef Link)

[11] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3512-3524, 2018. Article (CrossRef Link)

[12] Y. Liu, S. Wang, Q. Zhao, S. Du, A. Zhou, X. Ma, and F. Yang, "Dependency-aware task scheduling in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp.4961-4971, 2020. Article (CrossRef Link)

[13] Y. Chen, Y. Zhang, Y. Wu, L. Qi, X. Chen, and X. Shen, "Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8419-8429, 2020. Article (CrossRef Link)

[14] H. Tan, Z. Han, X. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. of IEEE INFOCOM 2017*, Atlanta, GA, USA, pp. 1-9, 2017. Article (CrossRef Link)

[15] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702-3712, 2016. Article (CrossRef Link)

[16] Z. Han, H. Tan, X. Li, S. H.-C. Jiang, Y. Li, and F. C. M. Lau, "OnDisc: online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2472-2485, 2019. Article (CrossRef Link)

[17] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360-374, 2021. Article (CrossRef Link)

[18] T. Zhao, S. Zhou, X. Guo, et al., "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *Proc. of IEEE ICC 2017*, Paris, France, pp. 1-7, May 2017. Article (CrossRef Link)

[19] Y. Deng, Z. Chen, X. Yao, S. Hassan, and J. Wu, "Task scheduling for smart city applications based on multi-server mobile edge computing," *IEEE Access*, vol. 7, pp. 14410-14421, 2019. Article (CrossRef Link)

[20] X. Chen, N. Thomas, T. Zhan, and J. Ding, "A hybrid task scheduling scheme for heterogeneous vehicular edge systems," *IEEE Access*, vol. 7, pp. 117088-117099, 2019. Article (CrossRef Link)

[21] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Generation Computer Systems*, vol. 85, pp. 1-8, 2018. Article (CrossRef Link)

[22] M. Zhao, W. Wang, Y. Wang, and Z. Zhang, "Load scheduling for distributed edge computing: a communication-computation trade off," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1418-1432, 2019. Article (CrossRef Link)

[23] C. Li, J. Bai, and J. Tang, "Joint optimization of data placement and scheduling for improving user experience in edge computing," *Journal of Parallel and Distributed Computing*, vol. 125, pp. 93-105, 2019. Article (CrossRef Link)

[24] Z. Lv, and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5706-5712, 2019. Article (CrossRef Link)

[25] L. Nkenyereye, L. Nkenyereye, S. M. R. Islam, C. A. Kerrache, M. Abdullah-Al-Wadud, and A. Alamri, "Software defined network-based multi-access edge framework for vehicular networks," *IEEE Access*, vol. 8, pp. 4220-4234, 2019. Article (CrossRef Link)

[26] C. Huang, M. Chiang, D. Dao, W. Su, S, Xu, and H. Zhou, "V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17741-17755, 2018. Article (CrossRef Link)

[27] F. Zeng, Y. Chen, L. Yao, and J. Wu, "A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing," *Peer-to-Peer Networking and Applications*, vol. 14, no. 2, pp. 467-481, 2021. Article (CrossRef Link)

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

[29] F. Zeng, R. Wang, and J. Wu, "How mobile contributors will interact with each other in mobile crowdsourcing with word of mouth mode," *IEEE Access*, vol. 7, pp. 14523-14536, 2019. Article (CrossRef Link)

[30] S. Mao, J. Wu, L. Liu, D. Lan, and A. Taherkordi, "Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing," *IEEE Systems Journal*, vol. 16, no. 1, pp. 287 - 298, 2022. Article (CrossRef Link)

[31] A. Mahmood, W. Zhang, and Q. Sheng, "Software-defined heterogeneous vehicular networking: The architectural design and open challenges," *Future Internet*, vol. 11, no. 3, pp. 70, 2019. Article (CrossRef Link)

[32] A. Mahmood, B. Butler, and B. Jennings, "Towards efficient network resource management in SDN-based heterogeneous vehicular networks," in *Proc. of 2018 IEEE 42nd Annual Computer Software and Applications Conference*, pp. 813-814, Jul. 2018. Article (CrossRef Link)

[33] S. R. Pokhrel, "Software defined Internet of vehicles for automation and orchestration," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3890-3899, 2021. Article (CrossRef Link)

[34] V. Nguyen, A. Brunstrom, K. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: a survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1567-1602, Apr. 2017. Article (CrossRef Link)

[35] Y. Mao, J. Zhang, and K. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. of IEEE WCNC 2017*, San Francisco, CA, USA, pp. 1-6, Mar. 2017. Article (CrossRef Link)

Mengmeng Sun received the M.S. degree in software engineering from Hunan Normal University, Hunan, China, in 2021. She is working in College of Cultural Communication, Henan Vocational Institute of Arts Zhengzhou, China. Her research interests include edge computing, software defined network, machine learning.

Lianming Zhang received his Ph.D. degree from the School of Information Science and Engineering of Central South University, China, and his M.Sc. and B.Sc. degrees from the Department of Physics of Hunan Normal University. He is currently a professor in the College of Information Science and Engineering of Hunan Normal University, China. His main research interests include network intelligence, software-defined networking, and edge computing.

Jing Mei received the Ph.D in computer science from Hunan University, China, in 2015. She is currently an assistant professor in the College of Information Science and Engineering in Hunan Normal University. Her research interests include parallel and distributed computing, cloud computing, edge computing etc.

Pingping Dong received her B.S., M.S. and Ph.D degree from the School of Information Science and Engineering at Central South University, China. Currently she is an associate professor in the college of information science and engineering, Hunan Normal University, China. Her research interests include protocol optimization and protocol design in wide area networks (WANs) and wireless local area networks (WLANs).