IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 December 2023, accepted 8 January 2024, date of publication 15 January 2024, date of current version 13 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354272

RESEARCH ARTICLE

Framework of Cloud Computing Resource Scheduling for Vehicle Fault Diagnosis

WANYI GU[1], HUA XU[1], AND LINA ZHU[2], (Member, IEEE)
[1]Information and Navigation College, Air Force Engineering University, Xi'an, Shaanxi 710043, China
[2]State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi 710071, China

Corresponding author: Lina Zhu (lnzhu@xidian.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61906156.

ABSTRACT Internet of Vehicles (IoVs) provides communication and computing resources, which makes the on-board diagnosis of vehicle faults possible. However, those resources need to be expanded to support the accurate analysis of the on-board diagnosis. Vehicular Cloud Computing (VCC) can solve the pressure of local vehicle processing but will cause an unavoidable delay. Thus, the accuracy and timeliness of on-board diagnosis cannot be guaranteed. To address the issue, we propose a Mobile Edge Caching based Resource Scheduling (MECRS) mechanism for the on-board diagnosis of vehicle faults. According to the urgency of vehicle fault diagnosis, we first design a cloud scheduling algorithm to meet the computation requirements of both the essential business of IoVs and the fault diagnosis. Subsequently, the priority allocation strategy is made for all four types of requests. Then, the urgent requests can be processed timely. Specifically, a multi-objective optimization method is proposed to allocate communication and computing resources for the above requests. In addition, we present a mobile edge caching algorithm in which the large-scale file with high popularity is offloaded to alleviate the pressure of the cloud. Finally, we carry out comprehensive simulations. The results reveal that the developed mechanism provides a high service rate for on-board diagnosis with limited network resources, while the performances of the other three essential services are not compromised.

INDEX TERMS Mobile edge caching, vehicle fault diagnosis, resource scheduling, vehicular cloud computing.

I. INTRODUCTION

With the increasing utilization of vehicles, various unexpected problems arise when the vehicle is in high-intensity overload operation. By 2023, the top three countries in terms of vehicle ownership are China, the United States and Japan. Among them, China ranks first with 430 million motor vehicles, 520 million drivers and 18.21 million new energy vehicles. The increase of vehicle ownership brings traffic problems such as traffic jams, vehicle accidents and environmental pollution. There are many kinds of problems in vehicles with long-term load operation. Therefore, efficient vehicle fault diagnosis is essential. However, the overdue vehicle fault diagnosis results in abnormal performance, even bringing security threats. Then, vehicles are damaged, and public safety is threatened [1], [2]. Accordingly, with a timely and accurate operation of vehicle fault diagnosis, a connected vehicle can receive fault warnings in time and ensure safety in efficiency.

Depending on the experience of experts, traditional fault diagnosis technologies could be more efficient and timely, such as the model-based approaches model-based approaches, signal-based approaches, and knowledge-based approaches [3], [4]. Then, with the development of computer technology, three main methods have gradually formed, signal processing, reliability statistics, and big data processed online [5], [6], [7]. Thanks to the development of the big data technology, vehicle fault diagnosis is currently faster and more efficient. Based on the big data technology, a vehicle need to use local computing resources to process massive data collected by sensors [8], [9]. However, more than local computing resources of vehicles is required. As the amount of data increases, the tasks of vehicle fault diagnosis must be processed promptly. This problem is

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

urgently solved to relieve the processing pressure of local resources.

The Vehicle-to-X(V2X) communication provided by IoVs [10] and vehicular cloud computing (VCC) technology [11] are considered to solve this problem. Utilizing the V2X communication, those tasks responded not in time could be offloaded to the vehicular cloud. The cloud platform has enormous storage capacity and computing power. Then, by uploading massive data related to those tasks, a vehicle could download the processed diagnosis results provided by VCC. The high mobility of IoVs induces intermittent connectivity, so uploading data and downloading results become difficult [12], [13]. The requests that the cloud cannot handle promptly should be discarded. Overall, there are two challenges to guarantee the timeliness and accuracy of vehicle fault diagnosis.

- Cloud computing, with its powerful storage and computing capabilities, has been applied to various researches. Currently, there is relatively little research in the intersection of cloud computing and vehicle fault diagnosis. How to schedule cloud resources to serve vehicle fault requests is challenging.
- Most researches on cloud resource allocation only focus on a single scheduling target and request [14], [15]. However, with diversified services, the cloud platform is for more than vehicle fault on-board diagnosis. It must maintain other essential services, such as network operation requests, emergency requests, and large-scale file requests. Hence, a scheduling mechanism for multitasking with comprehensive indicators has positive research significance [16].
- We hope the cloud platform can reserve more computing resources to serve vehicle fault requests. Then we need to reduce the resource footprint of other requests on the cloud, which may affect their services. How to maximize the service rate of vehicle fault requests on the premise of ensuring that other requests can be serviced normally is the research focus [17].

In this paper, we propose a Mobile Edge Caching based Resource Scheduling (MECRS) mechanism for vehicle fault on-board diagnosis. The purpose of this paper is to reserve more computing resources for handling vehicle fault requests while ensuring timely processing of the other three types of business. We first design a priority allocation strategy for all four types of requests. Next, a mobile edge caching algorithm is designed to process the large-scale file requests of the cloud. In addition, a multi-objective optimized resource scheduling method is proposed to allocate cloud computing resources. Finally, a comprehensive simulation is carried out to maximize the vehicle fault diagnosis service rate and make other requests receive timely responses. The significant contributions of this paper are as follows.

- A **multi-task scheduling mechanism**. Applying cloud computing to vehicle fault diagnosis cannot ignore the diversity of tasks on cloud platforms. According to the Quality of Service (QoS) requirements, we classify requests into four categories, targeting different applications. This not only ensures sufficient processing capability for vehicle fault requests, but also ensures timely response to other tasks.
- A **priority allocation strategy**. Accordingly, the vehicle fault diagnosis requests can be spited into two types, which are the emergence part processed locally, and delay tolerance part responded by cloud. The fault diagnosis with high timeliness and accuracy can be obtained. Moreover, we set an emergency factor for the emergency request in the cloud that maintains its priority. Finally, all requests uploaded to the cloud platform are sorted according to the priority policy.
- A **mobile edge caching algorithm**. In order to avoid the large-scale file requests taking up too much cloud computing resources, we formulate a vehicle popularity-based mobile caching model to make the cloud have more resources to serve the vehicle fault requests and ensure the timely service of large-scale file requests. In the model, we offload large file requests to the service vehicle for processing, and the files are cached in advance by the service vehicle when the communication channel is idle.
- A **global optimization method**. Based on the allocation strategy, a problem of optimizing resource allocation in the cloud is raised to maximize the system reward and minimize the large-scale file downloaded data volume from the cloud. Then, we propose a multi-objective optimization method to efficiently find sub-optimal solutions, thereby improving the service rate with low complexity.
- A **comprehensive simulation** is conducted, which validate our results from two aspects, the user side and the service side. Simulation results show that the vehicle on-board diagnosis requests serviced interval in 100 percent was elongated while ensuring the performances of others network basis services, and even under the premise of high vehicle density, we can guarantee the service rate of 50 percent. And for the set simulation scale, the optimal number of virtual machine configurations was given, which has practical reference value in reality.

The remainder of this paper is organized as follows. Section II reviews related works while Section III presents the system model. In section IV discuss the resources allocation for VCC and propose MERCS mechanism. Then, simulation results are given in Section V. Finally, we conclude the paper and look forward to future work in Section VI.

II. RELATED WORK

The driving safety of vehicles is highly related to the protection of life and property, so the fault diagnosis of vehicles has always been the focus in the industry and academia. At present, there are many mature algorithms suitable for fault diagnosis. For instance, in [18], a novel fault diagnosis procedure based on improved symplectic

geometry mode decomposition (SGMD) and optimized support vector machine (SVM) is presented, which is proved that the method is effective and robust for rotating machineries fault diagnosis. Aojia et al. design a fault detector and a fault-tolerant tracking controller. They propose a delay-dependent stability criterion to deal with the adverse effects of delay [19]. An active fault tolerant control scheme for an underwater remotely operated vehicle is proposed in [20], the authors implemented output control for a rod-operated vehicle. Literature [21] develops a fault diagnosis system for electric vehicle charging devices based on Fault Tree Analysis (FTA), which can accurately locate and quickly resolve the charging devices faults. Whatŕs more, Paper [22] proposes a soft SC fault diagnosis method based on the extended Kalman filter (EKF) for on-board applications in EVs. In the proposed method, the EKF estimates the faulty cell's state of charge (SOC) by adjusting a gain matrix based on real-time measured voltages. It is practical and robust in quickly detecting a soft SC fault. However, there are many types of vehicle faults. Therefore, a vehicle fault diagnosis mechanism for extensive fault data is urgently needed.

With the emergence of big data technology, scholars propose effective detection methods that can handle more data. Online diagnosis can ensure the real-time processing of vehicular fault tasks. In [2], a novel in-vehicle intelligent electric power supply network is proposed. This paper reveals that each device's power supply process is appropriately monitored, online faults are detected successfully, and the fault-tolerant method can realize remediation and protection in real-time. Zhang et al. [23] utilized BPNN for decision making and classification after conducting feature analysis and judgment. Liu et al. [24] designed a three-layer BP neural network structure to realize efficient fusion of Floating Car Data (FCD) and Fixed Detector Data (FDD). Tian et al. [25] presented a bearing fault detection method using KNN, which extracts fault features through spectral kurtosis and cross correlation.

Online diagnosis has the advantage of good timeliness. Still, the local processing of a large number of faulty services will put pressure on the local processor, and the processor also needs to process other IoV services. Barabino et al. design an offline framework for the diagnosis of time reliability by automatic vehicle location data. This framework can be adopted by transit managers for accurate reliability analysis [26]. Paper [27]introduced an intelligent diagnosis method based on a data-driven model for sensor intermittent fault. Paper [28] proposed a speed sensor fault diagnosis methodology based on a learning-based data-driven principle in induction motor drive systems. However, it is challenging to ensure timeliness by only relying on the offline diagnosis. In response to the need for more local resources, achieving a multi-objective joint diagnosis is impossible. We want a multi-objective optimization mechanism to ensure timeliness and accuracy. Therefore, We apply VCC to upload and download data by combining online and offline methods. We put the time-delay-sensitive requests on the local vehicle for online processing. The time-delay insensitive requests are uploaded to the cloud for offline processing, and the processing results are sent back to the vehicle through the cloud platform.

Many works on cloud computing have been carried out to enhance the service capabilities of the cloud. Moreover, reasonable resource scheduling can improve the service rate. The University of Ottawa scholars do research on virtual machine migration in cloud computing, thereby increasing resource utilization [29]. A multiple resource allocation approach was introduced in [30] that generalizes max-min fairness to multiple resource types. It formulated to maximize the minimal energy balance among all users by jointly optimizing time assignments. Zheng et al. propose an optimal computing resource allocation scheme to maximize the total expected reward of the connected vehicular cloud system. The scheme represents the optimization problem as an infinite-level semi-Markov decision process, which obtains significant performance gains within an acceptable complexity range [16]. Zhao et al. discuss the fair allocation of multiple resources and propose a new allocation mechanism called Dominant Resource with Bottlenecked Fairness (DRBF). They divide users into different queues based on their dominant resources to ensure that users in the same column receive an allocation according to their fair shares [31]. In [32], a contract model is adopted to address the cloud computing resource allocation and pricing problem in the mobile blockchain. and proposed a adverse selection contract solution to overcome the information asymmetry problem.

This paper focuses on the VCC architecture, which applies cloud computing to the IoV and then uploads fault tasks to the cloud. The volume of vehicle fault data is enormous. But requests in VCC are diverse, and the performances of other essential services cannot be compromised. Some common mechanisms, such as FCFS (First-Come-First-Served) mechanism, cause untimely service and huge resource waste. The vehicle fault diagnosis mechanism orienting towards multiple requests in VCC is closer to the actual scenario and solves the problem of poor communication and computing resources. Hence, an effective cloud resource allocation mechanism becomes the interest of this paper.

III. SYSTEM MODEL

This section describes the system model in the following sections. Firstly, we introduce the communication scenario. Then we describe the traffic model, overall optimization objective, system reward function model, and vehicle collaborative transmission model. Vehicles send various types of requests to the center base station, and base station forwards them to the cloud platform for processing. The processed results will be sent back to the vehicles. Due to the high dynamic topology of IoV, the dotted circles represent the effective communication range of the base station. Vehicles outside the range cannot communicate directly with the base station. Therefore, the V2I (Vehicle to Infrastructure) communication can be performed within the transmission range of the base

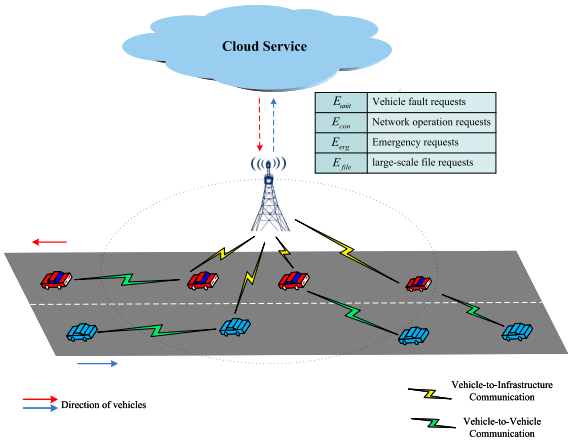


FIGURE 1. Scenarios model.

station, and the V2V (Vehicle to Vehicle) communication can be performed outside the transmission range.

We consider a vehicular communication scenario that occurred within a base station and the cloud platform, as shown in Fig.1, including the forward driving and reverse driving vehicles. The user vehicles sends four types of tasks to the cloud platform and forwards them by the base station. The flowchart is shown in Figure 2. It represents the process from the vehicle sending the requests to the cloud platform processing the requests. Besides, the Table 1 is the list of important notations in this paper.

A. TRAFFIC MODEL

According to the Quality of Service (QoS) requirements, we classify requests into four categories, including vehicle fault requests E_{unit} (focus on vehicles normal driving), network operation requests E_{con} (focus on network stability between the cloud platform and the vehicle), emergency requests E_{erg} (focus on cloud emergency treatment), and large-scale file requests E_{file} (focus on driving experience). The E_{unit}, the E_{con} and the E_{file} are made by the vehicles. The E_{erg} are generated by the cloud platform itself. After receiving the three kinds of requests, the base station transmits them to the cloud platform. Then the cloud platform computes the system reward, and the largest reward request has the highest priority.

B. OVERALL OPTIMIZATION OBJECTIVE

We choose the overall system reward as the optimization objective. And integrate the computing resources, bandwidth resources, energy consumption, and time consumption from the VCC into the reward function:

$$\begin{aligned} &\max R(S,O) && (P1)\\ &\min C_{file} && (P2)\\ &s.t.\ C1: C_{unit}+C_{con}+C_{erg}+C_{file} < C,\\ &\quad C2: C_{unit} > C_{file},\\ &\quad C3: \sum_{i=1}^{Kj} Ei \le c \qquad \forall j \in K, && (1) \end{aligned}$$

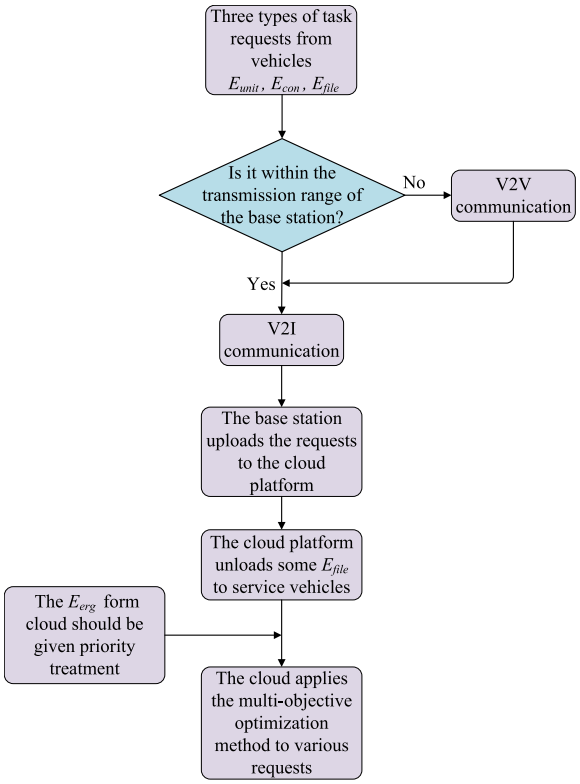


FIGURE 2. Scenarios flowchart.

TABLE 1. List of important notations.

ru_{Com}	Calculation rate of per virtual machine
ru_{Bw}	Occupied bandwidth of per virtual machine
ru^{P}_{Com}	Computation price of per virtual machine
ru^{P}_{Bw}	Bandwidth price of per virtual machine
β_c	Energy consumption unit price with no cloud
β_t	Time consumption unit price with no cloud
N_Q	The number of the requests served by cloud
N_{RU}	Maximum number of virtual machines in cloud
T_{ti}	User tolerance time for request i
T_i	The playout time required for large-scale request i
E_i	Data volume for request i
N_{ri}	Popularity degree for request i
E_{vi}	Energy consumption by no cloud for request i
T_{vi}	Time consumption by no cloud for request i
δ_i	Time from base station to cloud for request i
P_i	Transmission power from station to cloud for request i
Ω_i	Average received power in fading process for request i
r_v	Communication radius of per vehicle
v_{v1}	Speed of the target vehicle
v_{v2}	Speed of and the assisting vehicle
T_{chg}	time to maintain the vehicles relative motion
c	Cache capacity of per vehicle
λ	Transmission rate between target and auxiliary vehicle
$E[r_h]$	Download rate of vehicle
$E[r_p]$	Viewing playout rate of large-scale files
$E[B_i]$	Busy length of the playout buffer when request i arrives
$E[I_i]$	Free length of the playout buffer when request i arrives
$E[D]$	Contact duration between user vehicle and cache vehicle

where $R(S,O)$ is the system reward function. C is the maximum computing resource for the cloud. The total capacity of the four tasks together should be smaller than C. The c

is the capacity that the vehicle can cache large files. C_{unit}, C_{con}, C_{erg}, C_{file} are the computing resources occupied by vehicle fault requests, network operation requests, emergency requests, and large-scale file requests respectively. $C2$ means C_{unit} should be as large as possible than C_{file}. $C3$ represents the per vehicle cache capacity that is limited. K is the set of vehicles, $K = \{1, 2, 3, \ldots, j \ldots\}$, K_j is the maximum number of large-scale file requests that the vehicle can hold.

This system aims to get the maximum system reward and allow more communication and computing resources for vehicle fault transmission and processing. However, reducing the C_{erg} will ignore the network emergencies, and decreasing the C_{con} will have a significant impact on network stability, which further result in poor communication quality. Furthermore, users' satisfaction with the driving experience dropped with lower C_{file}. However, the response to large-scale file requests E_{file} may not necessarily apply cloud resources. Therefore, we propose a mobile edge caching algorithm in section IV-B, which specifies that the large-scale file requests are served by vehicle caching, thereby minimizing large-scale file service from the cloud.

C. SYSTEM REWARD FUNCTION MODEL

The reward function can be obtained by the difference between the gain function $E(S, O)$ and the overhead function $P(S, O)$, as shown in Eq.(2):

$$R(S, O) = E(S, O) - P(S, O). \tag{2}$$

$S = \{s|s = \left(ru_{Com}, ru_{Bw}, ru^P_{Com}, ru^P_{Bw}, \beta_c, \beta_t, N_Q\right)\}$ is the state set of the VCC platform. $O = \{0, 1, 2, \ldots, N_{RU}\}$ is the operation set of the VCC platform. $B = \{R_a, R_l\}$ is the event set. R_a means request arrivals. R_a means request leaves.

We use $E(S, O)$ represents the gains in time and power consumption. Taking action O under state S, it can be expressed as follows:

$$E(S, O) = \sum_{i=1}^{N_Q} e_i(S, O), \tag{3}$$

where $e_i(S, O)$ means the gain of a single request processed by the cloud platform. As this paper considered the system, the benefits of all requests should be synthesized. Furthermore, since the cloud platform has powerful resources and data storage capacity, the main content of $e_i(S, O)$ is caused by time consumption and energy consumption, as shown in the following:

$$e_i(S, O) = \begin{cases} 0 & o_p = 0 \\ [\beta_c(E_{vi} - P_i\delta_i) + \beta_t(T_{vi} - \delta_i)] & o_p \neq 0. \end{cases} \tag{4}$$

When $o_p = 0$, the vehicle does not enjoy the cloud service, so the gain is zero. When $o_p \neq 0$, the vehicle enjoys the cloud service. Therefore it avoids the enormous energy consumption E_{vi} and the time consumption T_{vi} required by ordinary internet downloading. But the cloud platform process makes $P\delta_i$ and δ_i consumes added. Additionally, the consumptions in cloud platform processing are reflected in overhead function $P(S, O)$.

$P(S, O)$ is the system overhead function for the requests, including energy consumption, time consumption and energy consumption caused by the channel fading, which can be expressed as follows:

$$P(S, O) = C(S, O)\tau(S, O) + \sum_{i=1}^{N_Q} \beta_c(P_i - \Omega_i) \cdot \delta_i, \tag{5}$$

where $C(S, O)$ means the overhead incurred per unit of time. $\tau(S, O)$ is the time to serve the request by cloud. $\sum_{i=1}^{N_Q} \beta_c(P_i - \Omega_i) \cdot \delta_i$ is the overhead caused by channel fading. Moreover, since $C(S, O)$ is determined by the number of virtual machines allocated by the cloud platform, the following equation can be obtained:

$$C(S, O) = \sum_{i=1}^{N_{RU}} i \cdot n_i \cdot (ru_{Bw} \cdot ru^P_{Com} + ru^P_{Bw}), \tag{6}$$

where n_i means the number of requests arranged for i virtual machines, and $\sum_{i=1}^{N_{RU}} i \cdot n_i$ is the total number of virtual machines occupied for requestes.

We allocate enough virtual machine resources for requests which can be processed within the user's tolerance time, so the processing time for each request is

$$\tau(S, O) = \frac{D_{every}}{ru_{com}}, \tag{7}$$

where D_{every} is the data volume processed by per virtual machine.

D. VEHICLE COLLABORATIVE TRANSMISSION MODEL

The V2V communication method supplements the V2I communication method, which enables vehicles to get services outside the transmission range. It reduces the pressure on communication and computation services. Therefore, V2V becomes a powerful support for improving cloud services in VCC.

Here, we focus on the communication between two vehicles. V2V communication between multiple vehicles can be achieved by multi hop transmission. If vehicle A transmits a request from vehicle B by V2V, the vehicles A and B are the target vehicle and the assisting vehicle, respectively. According to the vehicle traffic scene, V2V communication can be divided into three situations: the target vehicle and the assisting vehicle are relative driving, opposite driving, and the same direction driving. On the basis of the IEEE 802.11p agreement, the transmission bandwidth of V2V is 100kB/s, and the transmission radius of vehicle

communication equipment transmission is [50m, 500m].

$$T_{v2v} = \begin{cases} \dfrac{2r_v}{v_{v1} + v_{v2}} & v_{v1} \cdot v_{v2} \leq 0 \\ \dfrac{2r_v}{|v_{v1} - v_{v2}|} & v_{v1} \cdot v_{v2} > 0, v_{v1} \neq v_{v2} \\ \min\{\dfrac{E_i}{\lambda}, T_{chg}\} & v_{v1} = v_{v2}, \end{cases} \tag{8}$$

The first segment indicates the communication duration when the vehicles are in relative driving, including driving relative and opposite driving. Furthermore, the communication range tangent is the maximum communication distance between two vehicles. The second segment expresses the communication duration for two vehicles traveling in the same direction but not relatively stationary, and the communication duration for the relatively stationary vehicles is showed in the third segment. At this time, the communication between the two vehicles can be maintained until the request is completed or the relative motion changes. In addition, this model is also suitable for vehicles to download large-scale files from the cloud.

IV. RESOURCE ALLOCATION FOR VCC

To solve the optimization problem of the objective function, we propose a Mobile Edge Caching based Resource Scheduling (MECRS) mechanism. Its content includes the following three parts.

A. PRIORITY ALLOCATION STRATEGY

Due to lacking local computing resources, we upload some vehicle fault requests to the cloud platform. According to the user tolerance time, data volume, and popularity of requests, the cloud platform prioritizes the four types of requests. This method is mainly divided into the following three parts:

(1) Local computing

For the E_{unit}, according to the relationship between the service tolerance time T_t and the time threshold T_s, we can choose local computing or cloud computing.

When $T_t < T_s$, it belongs to the emergency vehicle fault request. At this point, the requests should be processed in the vehicle locally to ensure timely processing.

When $T_t > T_s$, this request is not time-sensitive. In this case, it should be uploaded services to the cloud for processing.

(2) Emergency response

Except for the local computing part, all services must occupy computing resources. Among the requests, E_{erg} should be responded to by cloud firstly due to their urgency. So, we set an emergency factor ζ for that. When E_{erg} occur, $\zeta = 1$, ensuring the preferential processing.

$$\zeta = \begin{cases} 1 & E_{erg} \ \ occurs \\ 0 & no \ \ E_{erg} \ \ occurs. \end{cases} \tag{9}$$

(3) Priority computing

Enough cloud computing resources are required for all requests uploaded to the cloud. Due to the rapid mobility of vehicle nodes, if we only follow the principle of first come, first served, we will miss out on some high reward requests. Therefore, we design priority rules. The priority equation is as follows:

$$\begin{aligned} P_{ri} &= (\omega_1/T_{ti} + \omega_2/E_i) \times N_{ri} \\ s.t.\ &\omega_1 + \omega 2 = 1, \end{aligned} \tag{10}$$

where ω_1 and ω_2 are the weights of user tolerance time and data volume, respectively. Additionally, the resource requirements for different requests are different. For example, the computation intensive files require virtual machine memory capacity and processing speed, and the communication intensive files focus on the need for network bandwidth resources. So, ω_1 and ω_2 are used to adjust the degree of emphasis on computing or communication resources. It can be seen from Eq. (17) that under the same conditions, the shorter user tolerance time, the smaller request data volume, and the higher request popularity, the higher the processing priority. This request is highly popular when the cloud platform receives multiple identical requests (such as the exact vehicle part fault, downloading the same large file, etc.). In the three priority considerations, the request popularity degree N_{ri} is the most important because when the cloud receives the request, it checks whether there are exact requests before making the scheduling decision. If the highest popularity request is selected and served, other identical requests can be obtained from V2V.

B. MOBILE EDGE CACHING ALGORITHM

As mentioned above, we divide the cloud requests into four categories. We aim to provide timely service for vehicle fault diagnosis requests with limited network and computing resources. In addition to the locally processed requests, we hope more cloud resources can be used to process the E_{unit}. However, the cloud platform cannot target one type of request. How do we ensure the other three types of requests response is the problem to solve. Reducing the C_{erg} will ignore the network emergencies, and decreasing the C_{con} will have a significant impact on network stability, which further result in poor communication quality. These two types of requests have to be handled in the cloud. Therefore, aimed at E_{file}, we propose a mobile edge caching algorithm, thereby minimizing the service of large-scale files from the cloud.

$$\begin{aligned} &\min C_{file} \\ &s.t. \sum_{i=1}^{K_j} E_i \leq c \qquad \forall j \in K. \end{aligned} \tag{11}$$

We divide vehicles into requesting vehicles and service vehicles. Of course, each vehicle can be either a requester or a server. We cache large-scale files into the service vehicle in advance based on specific rules. Using the V2V transmission method, the requesting vehicle can directly download large-scale files from the service vehicle, reducing the service of large-scale files from the cloud. This method

can ensure the service rate of E_{file}, reduce the computing pressure of the cloud platform, and leave more computing resources for E_{file}.

We choose the edge caching method to serve large-scale file requests in the resource allocation mechanism [33], because caching content at the edge of mobile networks is a promising way to solve data tsunami [29], [34]. It is assumed that when different vehicles have the same storage content, the overlap in time is small. This means that vehicles are storing the same content. The user vehicle only needs to send a request to a service vehicle until it is interrupted due to the transmission distance. In the resource buffer pool, this paper queues according to M/D/1. The service rate is $E[r_p]$, the file playback rate. The transmissive large file data size is $E[Y]$, and the arrival rate is λ_p.

$$\lambda_p = \lambda \cdot x_i, \tag{12}$$

$$E[Y] = E[D] \cdot E[r_h], \tag{13}$$

where $E[D]$ represents the transmission time between user vehicles and service vehicles, $E[r_h]$ represents the download rate of vehicles, λ represents the communication speed between the target vehicle and the auxiliary vehicle, that is, the task arrival rate, and x_i represents the number of the same files downloaded from the collaborative vehicle at the same time. It can be seen that the service rate of the queue:

$$\rho = \lambda_p \cdot \frac{E[Y]}{E[r_p]} = \lambda \cdot x_i \cdot E[D] \cdot \frac{E[r_h]}{E[r_p]}, \tag{14}$$

where ρ is the long term utilization of the queue, $E[r_p]$ is the viewing playout rate of large-scale files

When the large-scale file request i arrives, P_i is the free probability of the queue.

$$P_i = \frac{E[I_i]}{E[B_i] + E[I_i]} = 1 - \rho, \tag{15}$$

where $E[I_i]$ represents the free length of the playout buffer when request i arrives, and $E[B_i]$ represents the busy length of the playout buffer when request i arrives.

Because our model serves large files, the transfer time is extended. Then remember that C_{file} equals the number of total bytes downloaded from the cloud. We have that:

$$\lim_{T_i \to \infty} \frac{C_{file}}{T_i} = \frac{E[I_i] \cdot E[r_p]}{E[B_i] + E[I_i]}, \tag{16}$$

$$Ti = \frac{E_i}{E[r_p]}. \tag{17}$$

From the above derivation, we can get the following formula:

$$C_{file} = E_i \cdot (1 - \lambda \cdot E[D] \cdot \frac{E[r_h]}{E[r_p]}). \tag{18}$$

As can be seen from the above formula, when the other parameters are determined, C_{file} mainly depends on the characteristic of the file itself, which is the E_i. However, we focus on large-scale file requests, and large E_i cannot avoid them. Therefore, we research another file characteristic, the document-popularity. Popularity refers to the demand for documents. High popularity means multiple users can download such large-scale files. In section V, we propose a popularity-based greedy algorithm.

C. GLOBAL OPTIMIZATION METHOD

Based on the allocation strategy, a problem of optimizing resource allocation in the cloud is raised to maximize the system reward and minimize the large-scale file downloaded data volume from the cloud. Then, we propose a multi-objective optimization method to efficiently find sub-optimal solutions, thereby improving the service rate with low complexity.

As shown in algorithm 1, P1 is getting the maximum system reward. P2 is minimizing computing resources occupied by large-scale files. Both P1 and P2 have high computational complexity. Thus, we propose a greedy two-phase resource allocation algorithm to efficiently find sub-optimal solutions for Problem 1 and Problem 2 with low complexity. As depicted in Algorithm 1, Phase 1 (Algorithm 2) and Phase 2 (Algorithm 3) correspond to solving P1 and P2, respectively, as shown in algorithm 1.

Algorithms 2 and 3 are the decomposition of Algorithm 1. Algorithm 2 sorts requests based on priority and system reward functions, and utilizes greedy algorithms to allocate virtual machine resources to various types of requests. Algorithm 3 greedily caches the large file requirements of the user's vehicle into the service vehicle based on the file popularity. The sorting method of the algorithm is bubble sorting, which has a lower time complexity compared to fast sorting and other methods. In addition, the advantages of the greedy algorithm are its simplicity and low time complexity.

This paper focuses on effectively allocating the existing cloud computing resources, so that each optimization result can simplify the problem to a smaller sub-problem. In addition, considering the computing load and limited bandwidth of the cloud platform, and different tolerance time, data volume and popularity for various requests, if the requests are served according to the FCFS mechanism, computing resources can not be effectively utilized.

Given the simplicity and low time complexity of the greedy algorithm, this paper improves the request priority when selecting the optimal solution of sub-problems to make the resource allocation more comprehensive and reasonable. The algorithm simplifies the cloud platform resource optimization problem into some sub-selection problems. Simultaneously, each sub-selection maximizes the current benefit, so we obtain a local optimal solution. When the cloud platform has new resources released, the selection is made again to obtain the optimal local resolution of the new sub-problem. Furthermore, the repetition process can describe the vehicle continuously entering the base station communication range and sending requests in the actual scenario. Therefore, using a priority-based dynamic greedy algorithm can effectively optimize the cloud platform's computing resources and bandwidth resources to maximize the system reward.

Algorithm 1 Two-Phase Greedy Resource Allocation Algorithm

1) **For** all $j \in K$ **Do**
2) Phase 1: Execute Algorithm 2 to solve P1
3) Phase 2: Execute Algorithm 3 to solve P2
4) Obtain resource allocation solution
5) **End For**

Algorithm 2 Priority-Based Dynamic Greedy Algorithm

Input: I [*num_request*]
Output: Sorted I [*max_num_request*]
Initialization:

1) **Begin**
2) Initialize parameters, $K = \{1, 2, 3, \ldots, j \ldots\}$;
3) **For** $j \in K$ to *num_request*
4) Calculate corresponding $R(S, O)$, P_{ri};
5) **End For**
6) **For** $i = 1$ to *num_request* $- 1$
7) **If** $R(S, O)_i < R(S, O)_{i+1}$ **Then**
8) Exchange $I[i]$ and $I[i + 1]$;
9) **End If**
10) **If** $R(S, O)_i == R(S, O)_{i+1}$ **Then**
11) **If** $P_{ri} < P_{r(i+1)}$ **Then**
12) Exchange $I[i]$ and $I[i + 1]$
13) **End If**
14) **End If**
15) **End For**
16) Allocate VMs in the sequence of I [*max_num_request*]
17) **End**

Specifically, the algorithm first calculates the system rewards for each request the cloud platform receives. Next, sort by the reward level. If the requests have the same reward, according to the specified priority from high to low, thereby forming an overall ordered task queue. Then, starting from the first position of the task queue, each time trying to allocate enough virtual machines to meet the demand quantity in bandwidth and calculation. The allocation is completed if the optimization measure can be better than other requests. Otherwise, the next request in the task queue could be selected. The idea of the algorithm is shown below in algorithm 2.

The cloud platform collects requests in waiting period, and makes decisions during the decision period. The pseudo code represents a single resource scheduling scheme of the VCC during decision period, and the cloud platform always provides service to large-reward and high-priority requests. After collecting the service requests, the process is performed again in the next decision period. Since the time interval is small in adjacent periods, a dynamic cloud platform resource scheduling is formed. In addition, the time span of both periods is small, therefore forming a dynamic cloud platform resource scheduling decision process. Furthermore, we use I [*num_request*] to indicate the set of requests collected by the cloud platform while waiting for the decision, I [*max_num_request*] indicates the arranged request set, K is the set of vehicles.

Algorithm 3 Popularity-Based Caching Greedy Algorithm

Input: I [*file_request*], c
Output: Sorted I [*file_max_popular*]
Initialization:

1) **Begin**
2) Initialize parameters,;
3) **For** $j \in K$ to *file_request*
4) **If** $N_{r[j]} < N_{r[j+1]}$ **Then**
5) Exchange $I[j]$ and $I[j + 1]$;
6) **End If**
7) **End For**
8) Vehicle cache in the sequence of I [*file_max_popular*]
9) **For** $j = 1$ to *file_request*
10) **while** $\sum_1^j E_{vi[i]} + E_{vi[most\_popular]} < c$
11) $j \leftarrow j + 1$
12) **End For**
13) **End**

To solve the P2, we propose a popularity-based caching greedy algorithm in Algorithm 3. This algorithm replicates the most popular content in every service vehicle. Then, the vehicle's remaining storage is greedily filled with large-scale files on the basis of popularity from high to low. This way, the large-scale file requests are effectively offloaded to the service vehicle. Therefore, cloud resources' pressure to process multiple requests can be alleviated. Consequently, more computing resources are left for E_{file}, thereby improving the E_{file} service rate. In addition, $N_{r[i]}$ indicates the E_{file} popularity level, $E_{vi[i]}$ indicates the file size, and c is the capacity that the vehicle can cache large files.

V. SIMULATION AND ANALYSIS

In this section, we carry out comprehensive simulations. There are four service rates of successful requests as indicators in the evaluation, and the performance of the proposed mechanism MECRS is shown from two aspects: the user side and the service side. We first give the variation of the four requests service rates with vehicle density and then compare the rates under different requests proportion. Finally, the service rate changes with cloud virtual machine density show MECRS optimality.

A. SIMULATION SETUP

We simulate the communication scenarios as shown in Fig. 1, where N vehicles are within the coverage area of one cellular base station, and M vehicles are outside the range of the base station. The road includes a single lane in two directions, and vehicles are divided into relative and opposite driving. The user vehicles sends three types of requests to the cloud platform and forwards them by the base station. The other requests E_{erg} are generated by the cloud platform itself. The

TABLE 2. Simulation parameters.

Parameters	Value
Calculation rate of per virtual machine	3000 MIPS
Occupied bandwidth of per virtual machine	40 Mbps
Computation price of per virtual machine	1.1 yuan/hour
Bandwidth price of per virtual machine	0.8 yuan/hour
Energy consumption unit price with no cloud	1.2 yuan/hour
Time consumption unit price with no cloud	0.9 yuan/hour
The number of virtual machines	[0, 500]
The number of vehicles	[0, 400]
Transmission power from base station to cloud	100 W
Average received power in the fading process	20 W
Data volume processed by per virtual machine	6000 bytes

simulation analysis work is carried on MATLAB. We build a simulation tool to simulate the requests for vehicles to the cloud. We set the road length at 10,000 meters and put infrastructure at 5,000 meters in the middle. Furthermore, the effective communication range is 500 meters. The safe distance between each vehicle is 50 meters. The acceleration of the vehicle is typically distributed. Besides, real-time speed adjustment is carried out when the distance between vehicles is less than the safe distance.

As for the parameters of the network environment, we apply the IEEE 802.11p protocol, in which the transmission bandwidth of V2V is 100kB/s and the transmission radius of onboard communication equipment is [50m, 500m]. Here, We assume that large files are rated on a scale of popularity from 1 to 10, with 10 being the most popular and 1 being the least popular. Then, We set the scenario that the cloud platform has yet to reach saturation, which always remains computing resources. Simulation settings are shown in Table 2. The price of per virtual machine refers to Amazon's public cloud service price.

The simulation experiment is divided into two parts. This paper uses four request success service rate as evaluation metrics to demonstrate the performance of the MECRS mechanism from both the user and server perspectives. We define the service rate as the proportion of successfully responded requests to the total number of requests.

In the user-side experiment analysis, we first compare the trend of successful service ratios for four different services under different mechanisms as vehicle density increased. Then, we present the changes in service ratios for four different requests with different proportions. Finally, through the collected real data, we verify the effectiveness of the cloud platform assisted processing of vehicle fault diagnosis. In the server-side experimental analysis, we compare the occupancy of virtual machines under different mechanisms from computing resources, as the vehicle density increased. The optimal number of virtual machines is provided to avoid resource waste.

B. COMPARE MECHANISM

We consider and compare the performance with the following allocation policies:

- **MECRS**: This mechanism proposed in this paper divides local processing and cloud processing to use local resources fully. Then a multi-objective optimized resources scheduling method is applied.
- **Random**: All vehicle fault requests are uploaded to the cloud for processing, and the vehicle randomly caches large-scale files.
- **No-Caching**: All vehicle fault requests are uploaded to the cloud for processing, and the vehicle edge caching mechanism is not applied.
- **FCFS**: The cloud responds to all requests using the first come, first serve rule.

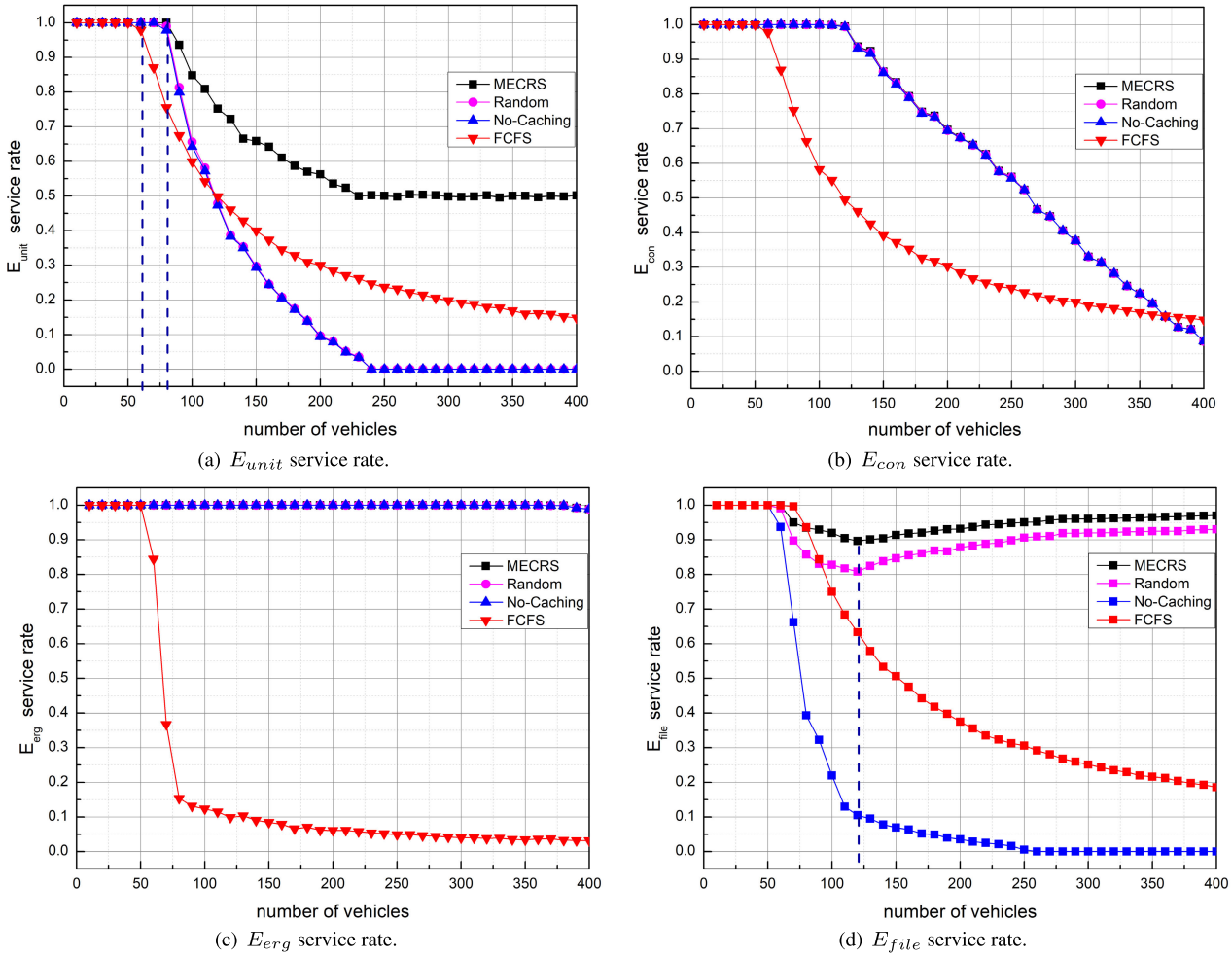
C. ANALYSIS FOR USER SIDE

We make each vehicle send requests every 1 second. Vehicles outside the effective communication range can transmit over multiple hops by V2V. The number of cloud VMs (Virtual Machines) is 200, and the proportion of four requests is 5:2:2:5. We assume that E_{unit} is proportional to E_{file}, and E_{erg} is proportional to E_{con}, Obviously, E_{unit} and E_{file} are sent more frequently. The objective function of this paper is to maximize the system reward function, which is mainly determined by gain function and the overhead function in Eq.(2). In addition, this paper assumes that the loss caused by channel fading is not considered.

It can be seen from Fig.3 that while the requests E_{con} and E_{erg} are served timely, the first three mechanisms can lengthen the interval that E_{unit} are 100 percent serviced. However, with the increase of vehicle density, under the Random and No-Caching mechanisms, E_{unit} are not allocated enough virtual machines due to the higher priority of category E_{erg} and E_{con}. Therefore, the vehicle fault service rate is even lower than that in the FCFS mechanism (the reason for the lower E_{file} service rate of the No-Caching mechanism in Fig.3(d) is identical). Then, we propose a priority-based algorithm and a popularity-based caching greedy algorithm to solve this problem. Fig.3(a) shows the MECRS mechanism can effectively improve the service rate of E_{file} compared to other mechanisms. Sequentially, even under the premise of high vehicle density, we can guarantee the service rate of 50 percent.

In Fig.3(b)(c), the emergency factor we set enables E_{erg} to be responded preferentially. Under the premise of 100 percent E_{erg} service rate, the E_{con} also has higher service rate to ensure network stability and security.

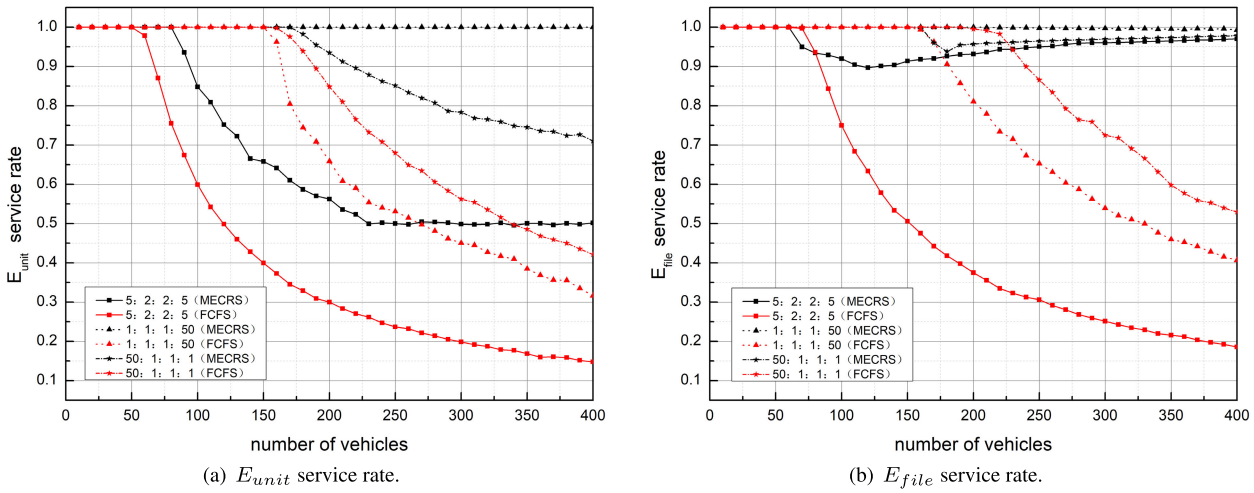
From Fig.3(d), we can see the MECRS mechanism improves the E_{file} service rate. When N=120 (the number of vehicles), there is minimum service rate. This is because the cloud platform numerous resources occupied for other higher priority requests, and the number of vehicles limits the capacity of cached files. Therefore, the lowest value appears. As the vehicle density increases, the total buffer capacity increases, so the service rate enhances and then level off. At this point, the service rate of the user getting large-scale files from the cache vehicles has reached the maximum, and the limiting factor becomes the cloud resources.



(a) E_{unit} service rate. (b) E_{con} service rate.

(c) E_{erg} service rate. (d) E_{file} service rate.

FIGURE 3. Service rate of four requests vs. number of vehicles, VMs = 200, proportion of four types is 5:2:2:5.



(a) E_{unit} service rate. (b) E_{file} service rate.

FIGURE 4. Service rate in different proportions vs. number of vehicles, VMs = 200.

Fig.4 shows the service rates of E_{unit} and E_{file} at different ratios. The first is both in the same proportion. The second is E_{file} weight ratio. The third is E_{unit} weight ratio. Under the 1: 1: 1: 50 ratio, due to the reduction of E_{unit}, the local processing capacity is sufficient and can reach a service rate of 100 percent. At the same time, with the increase in E_{file}, the

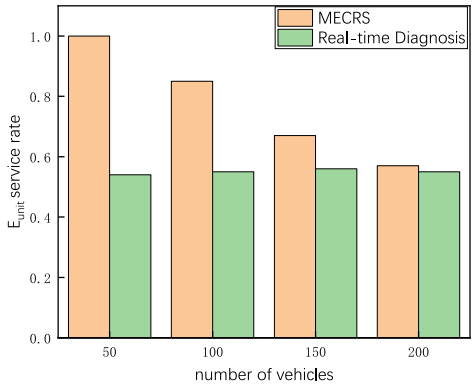


FIGURE 5. Service rate under different mechanisms.

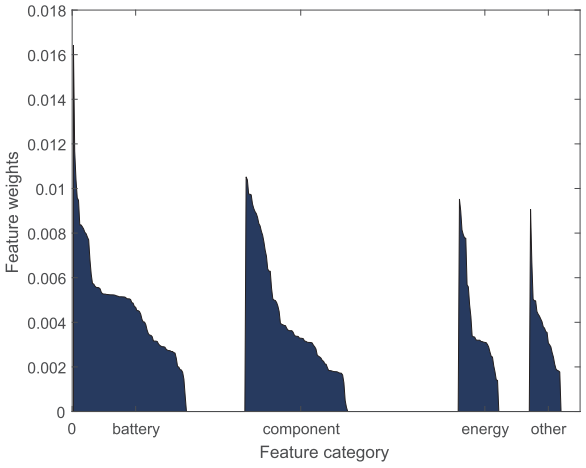


FIGURE 6. Feature weights.

vehicle cache files have been fully utilized so that the service rate can reach 99 percent. The remaining 1 percent is because there is no such file in the surrounding vehicles. This file must be downloaded from the cloud, which cannot be transmitted successfully within the tolerable time, resulting in a service failure. Furthermore, under the ratio of 50: 1: 1: 1, the E_{unit} service rate and the successful service rate have increased due to the increase in the request density. Meanwhile, the E_{file} decrease and there is less competition with sufficient cache capacity, improving the service rate. In summary, the proposed mechanism can increase the service request rate under various request ratios.

Traditional vehicle fault diagnosis algorithms mostly collect fault data and process it online. This can ensure the timeliness of task processing, however it will lead to insufficient computing capability of the local processor in vehicles. Especially when there are many other types of tasks that require local processing in vehicles, such as road section warnings, autonomous driving, etc. Therefore, this paper proposes a fault task offloading mechanism, keeping part of the vehicle failure tasks locally and offloading part of them to the cloud platform for processing. Details are elaborated in Section IV. A.

Literature [37] proposes a vehicle-mounted fault diagnosis system with low computational complexity and small data

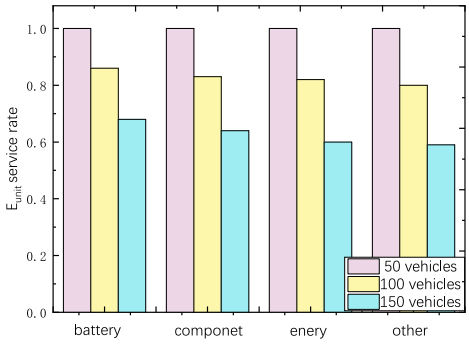


FIGURE 7. Service rate of vehicle features.

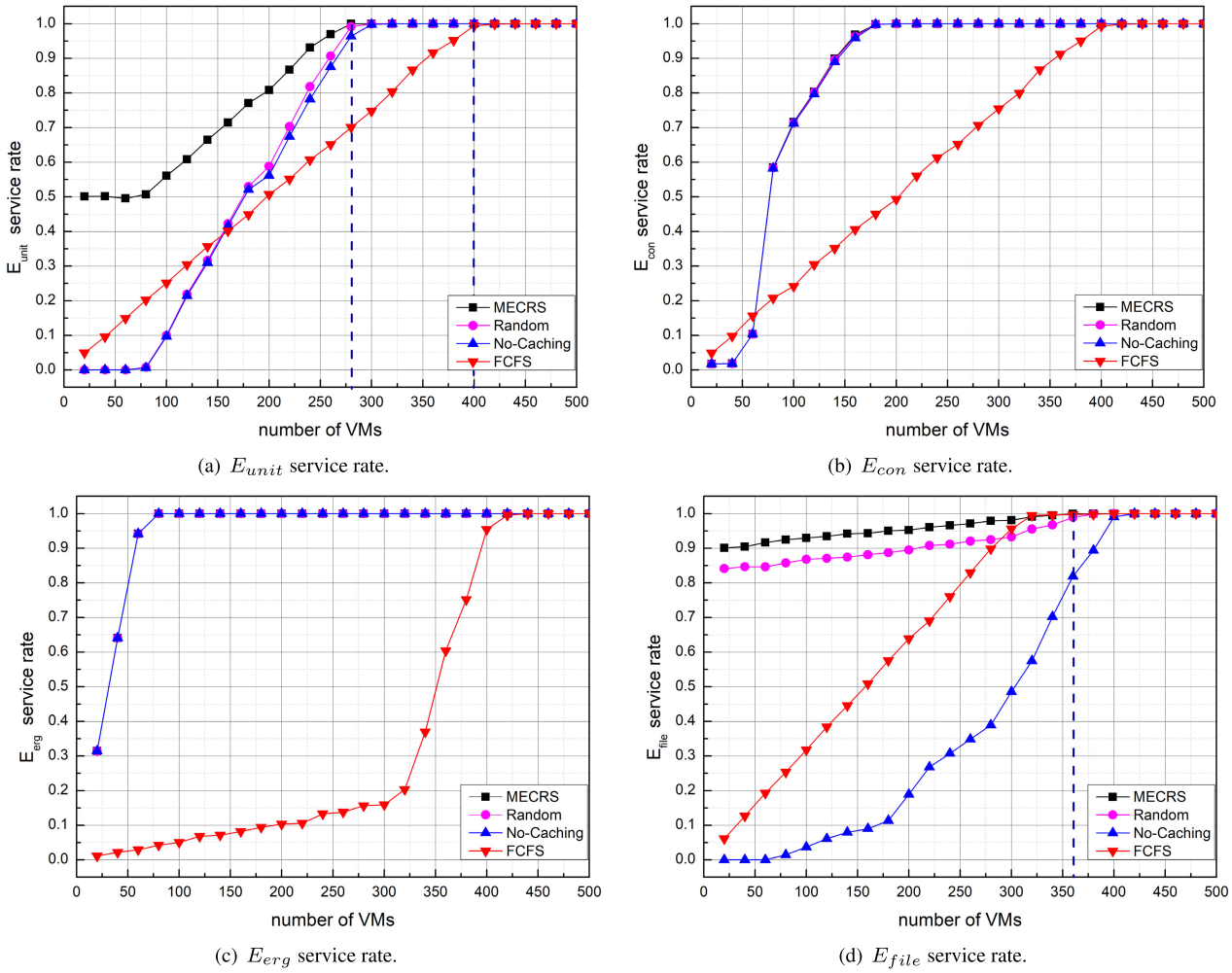
storage, for achieving real-time monitoring of vehicle status. It validates the effectiveness of the proposed diagnosis system in terms of accuracy, complexity and storage capacity. This paper obtained 466688 samples through Electronic Vehicle management, of which 177261 samples are faulty. In summary, there are 539 original features and the actual data volume is 6.52 GB. During the dataset collection process, it is inevitable to encounter some data errors and omissions. Therefore, it is necessary to preprocess the data before conducting simulation verification. The experimental results obtained from effective real data are more valuable and instructive. It mainly includes the following three steps: data integration, remove useless attributes and missing value processing.

When the local computing capacity is insufficient or largely occupied by other tasks, some fault tasks with high latency requirements may not be processed in time. At this point, the vehicle has driven out of the V2I transmission range, and also fails to transmit by V2V, which results the waste of resources. As shown in the simulation results in Fig. 5, assuming that only 20 percent of the vehicle CPU resources are occupied, it can be seen that when local resources are insufficient, the MECRS mechanism can improve the service rate of E_{file}. And it gradually decreases as the vehicle density increases. Moreover, the real-time diagnosis mechanism in [37] is not affected by vehicle density.

Fig. 6 shows the weight of each feature. we can see that features related to battery have the highest weights, and the second most important features are the one related to vehicle component. The third are the energy features. The remaining features are collectively referred to as the other features. We applied these four types of features to validate the mechanism proposed in this paper. As can be seen from Fig. 7, all four types of failure data received effective responses. When the vehicle density is low, 100 percent service rate can be achieved.

D. ANALYSIS FOR SERVICE SIDE

We make each vehicle send requests every 1s. The number of vehicles $N = 200$, and the proportion of four requests: 5:2:2:5.



(a) E_{unit} service rate.

(b) E_{con} service rate.

(c) E_{erg} service rate.

(d) E_{file} service rate.

FIGURE 8. Service rate of four requests vs. number of VMs, N = 200, proportion of four requests is 5:2:2:5.

From Fig.8, we can see that as the virtual machine density increases, the service rates of four requests improve. When virtual machine resources are insufficient, high priority E_{con} and E_{erg} occupy most virtual machines under the Random and No-Caching mechanisms, making E_{file} service rate even lower than that under FCFS mechanism. However, it improves significantly when the virtual machine number increases to 160. Under the E_{con} and E_{erg} responded in time, we effectively increased the service rate of E_{unit}. On the one hand, we ensure the requests are 100 percent served under the premise that the VMs is not saturated (280-400). On the other hand, when the VMs is insufficient (20-80), 50 percent service rate can also be guaranteed due to the vehicle local processing. In Fig.8(d), the E_{file} service rate under the No-caching mechanism is the lowest because the other three requests have higher priority than large-scale file requests. Hence, we use the Popularity-based edge caching mechanism to solve the problem of low file service rate. The proposed mechanism MRCRS obviously improves the E_{file} service rate, and the other two requests are also responded in time.

In addition, it can be observed that when all requests are 100 percent serviced, the optimal number of virtual machines is 360. However, at 260 units, the first three types of requests have reached 100 percent service rate. Between 260 and 360, the E_{file} service rate does not increase significantly. Therefore, from an economic perspective, the optimal number of virtual machines on the cloud platform is about 260, which has practical reference value in reality.

VI. CONCLUSION AND FUTURE WORD

In this paper, we proposed a computing resource allocation mechanism for the on-board vehicle fault diagnosis. Based on the tolerance time of diagnosis service, we first designed a priority allocation strategy for all four types of requests. Then, considering the QoS characteristics of requests, we made a priority allocation strategy to maximize the long-term reward of the VCC system. In addition, a mobile edge caching algorithm with high popularity was proposed which offloaded the large-scale file requests to alleviate the pressure of the cloud. Consequently, the

multi-objective optimization resource scheduling method was proposed to allocate communication and computing resources. Finally, the simulation results revealed that the E_{unit} 100 percent serviced interval was elongated while ensuring the timeliness of the other requests, and even under the premise of high vehicle density, we can guarantee the service rate of 50 percent. Compared with the real-time diagnosis mechanism, it can improve the service rate under the insufficient resources. This will significantly reduce the traffic impact caused by vehicle fault. And, from an economic perspective, for the set simulation scale, the optimal number of virtual machine configurations was given in 260, which has practical reference value in reality.

The research on vehicle fault diagnosis is very meaningful and can effectively avoid various traffic problems, such as traffic congestion, travel plans, traffic accidents, etc. By utilizing knowledge graph technology, the collected data can be used for knowledge association and mining, forming a huge vehicle fault knowledge base. In addition, the powerful computing and storage capabilities of cloud computing can more effectively solve vehicle fault diagnosis problems, which is a new direction that can be explored in the future.

REFERENCES

[1] Q. Shi and H. Zhang, "Fault diagnosis of an autonomous vehicle with an improved SVM algorithm subject to unbalanced datasets," *IEEE Trans. Ind. Electron.*, vol. 68, no. 7, pp. 6248–6256, Jul. 2021.

[2] W. Kong, Y. Luo, Z. Qin, Y. Qi, and X. Lian, "Comprehensive fault diagnosis and fault-tolerant protection of in-vehicle intelligent electric power supply network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10453–10464, Nov. 2019.

[3] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3757–3767, Jun. 2015.

[4] Y. Fang, H. Min, W. Wang, Z. Xu, and X. Zhao, "A fault detection and diagnosis system for autonomous vehicles based on hybrid approaches," *IEEE Sensors J.*, vol. 20, no. 16, pp. 9359–9371, Aug. 2020.

[5] A. Bondyra, P. Gasior, S. Gardecki, and A. Kasinski, "Fault diagnosis and condition monitoring of UAV rotor using signal processing," in *Proc. Signal Process., Algorithms, Architectures, Arrangements, Appl. (SPA)*, Sep. 2017, pp. 233–238.

[6] R. Dorociak, "Early probabilistic reliability analysis of mechatronic systems," in *Proc. Annu. Rel. Maintainability Symp.*, Jan. 2012, pp. 1–6.

[7] H. Zhang and W. Li, "A new method of sensor fault diagnosis based on a wavelet packet neural network for hybrid electric vehicles," in *Proc. 9th Int. Congr. Image Signal Process., Biomed. Eng. Informat. (CISP-BMEI)*, Oct. 2016, pp. 1143–1147.

[8] J. Lee, K. Oh, Y. Yoon, T. Song, T. Lee, and K. Yi, "Adaptive fault detection and emergency control of autonomous vehicles for fail-safe systems using a sliding mode approach," *IEEE Access*, vol. 10, pp. 27863–27880, 2022.

[9] Z.-H. Liu, J.-N. Li, W.-Y. Yang, M.-Y. Shen, X.-F. Shen, and Y. Luo, "An active fault detection for unmanned surface vehicles with minor fault," *IEEE Access*, vol. 10, pp. 119767–119776, 2022.

[10] L.-M. Ang, K. P. Seng, G. K. Ijemaru, and A. M. Zungeru, "Deployment of IoV for smart cities: Applications, architecture, and challenges," *IEEE Access*, vol. 7, pp. 6473–6492, 2019.

[11] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2244–2281, 3rd Quart., 2016.

[12] F. Cunha, L. Villas, A. Boukerche, G. Maia, A. Viana, R. A. F. Mini, and A. A. F. Loureiro, "Data communication in VANETs: Protocols, applications and challenges," *Ad Hoc Netw.*, vol. 44, pp. 90–103, Jul. 2016.

[13] X. He, H. Zhang, W. Shi, T. Luo, and N. C. Beaulieu, "Transmission capacity analysis for linear VANET under physical model," *China Commun.*, vol. 14, no. 3, pp. 97–107, Mar. 2017.

[14] A. B. M. B. Alam, M. Zulkernine, and A. Haque, "A reliability-based resource allocation approach for cloud computing," in *Proc. IEEE 7th Int. Symp. Cloud Service Comput. (SC)*, Nov. 2017, pp. 249–252.

[15] J. Chen, "A cloud resource allocation method supporting sudden and urgent demands," in *Proc. 6th Int. Conf. Adv. Cloud Big Data (CBD)*, Aug. 2018, pp. 66–70.

[16] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An SMDP-based resource allocation in vehicular cloud computing systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7920–7928, Dec. 2015.

[17] X. Wang, X. Wang, H. Che, K. Li, M. Huang, and C. Gao, "An intelligent economic approach for dynamic resource allocation in cloud services," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 275–289, Jul. 2015.

[18] X. Zhang, C. Li, X. Wang, and H. Wu, "A novel fault diagnosis procedure based on improved symplectic geometry mode decomposition and optimized SVM," *Measurement*, vol. 173, Mar. 2021, Art. no. 108644.

[19] M. Aojia, G. Feng, L. Yahui, Z. Lei, W. Chenlin, and L. Fugui, "Synergetic robust fault detection and fault tolerant control for flight vehicles with time-varying delay," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, May 2017, pp. 2594–2599.

[20] A. Baldini, R. Felicetti, A. Freddi, S. Longhi, A. Monteriù, and A. Fasano, "Fault detection, diagnosis and fault tolerant output control for a remotely operated vehicle," in *Proc. 14th IEEE/ASME Int. Conf. Mech. Embedded Syst. Appl. (MESA)*, Jul. 2018, pp. 1–7.

[21] D.-X. Gao, J.-J. Hou, K. Liang, and Q. Yang, "Fault diagnosis system for electric vehicle charging devices based on fault tree analysis," in *Proc. 37th Chin. Control Conf. (CCC)*, Jul. 2018, pp. 5055–5059.

[22] R. Yang, R. Xiong, and W. Shen, "On-board diagnosis of soft short circuit fault in lithium-ion battery packs for electric vehicles using an extended Kalman filter," *CSEE J. Power Energy Syst.*, vol. 8, no. 1, pp. 258–270, Jan. 2022.

[23] G. Jiang, H. He, J. Yan, and P. Xie, "Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3196–3207, Apr. 2019.

[24] J. Liu, J. Huang, R. Sun, H. Yu, and R. Xiao, "Data fusion for multi-source sensors using GA-PSO-BP neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 10, pp. 6583–6598, Oct. 2021.

[25] J. Tian, C. Morillo, M. H. Azarian, and M. Pecht, "Motor bearing fault detection using spectral kurtosis-based feature extraction coupled with K-nearest neighbor distance analysis," *IEEE Trans. Ind. Electron.*, vol. 63, no. 3, pp. 1793–1803, Mar. 2016.

[26] B. Barabino, M. Di Francesco, and S. Mozzoni, "An offline framework for the diagnosis of time reliability by automatic vehicle location data," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 583–594, Mar. 2017.

[27] K. Zhang, B. Gou, W. Xiong, and X. Feng, "An online diagnosis method for sensor intermittent fault based on data-driven model," *IEEE Trans. Power Electron.*, vol. 38, no. 3, pp. 2861–2865, Mar. 2023.

[28] Y. Xia, Y. Xu, B. Gou, and Q. Deng, "A learning-based method for speed sensor fault diagnosis of induction motor drive systems," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[29] T. K. Refaat, B. Kantarci, and H. T. Mouftah, "Dynamic virtual machine migration in a vehicular cloud," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2014, pp. 1–6.

[30] J. Liu, K. Xiong, D. W. K. Ng, P. Fan, Z. Zhong, and K. B. Letaief, "Max-min energy balance in wireless-powered hierarchical fog-cloud computing networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7064–7080, Nov. 2020.

[31] L. Zhao, M. Du, and L. Chen, "A new multi-resource allocation mechanism: A tradeoff between fairness and efficiency in cloud computing," *China Commun.*, vol. 15, no. 3, pp. 57–77, Mar. 2018.

[32] J. Wang, J. Li, Z. Gao, Z. Han, C. Qiu, and X. Wang, "Resource management and pricing for cloud computing based mobile blockchain with pooling," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 128–138, Jan. 2023.

[33] D. Liu and C. Yang, "Energy efficiency of downlink networks with caching at base stations," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 4, pp. 907–922, Apr. 2016.

[34] L. Vigneri, T. Spyropoulos, and C. Barakat, "Low cost video streaming through mobile edge caching: Modelling and optimization," *IEEE Trans. Mobile Comput.*, vol. 18, no. 6, pp. 1302–1315, Jun. 2019.

[35] M. Sathiamoorthy, A. G. Dimakis, B. Krishnamachari, and F. Bai, "Distributed storage codes reduce latency in vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 9, pp. 2016–2027, Sep. 2014.
[36] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1300–1313, Jun. 2018.
[37] L. Zhu, Y. Zhou, R. Jia, W. Gu, T. H. Luan, and M. Li, "Real-time fault diagnosis for EVs with multilabel feature selection and sliding window control," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18346–18359, Oct. 1, 2022.

WANYI GU received the B.Eng. degree in telecommunication engineering from the Hebei University of Technology, Tianjin, China, in 2018, and the M.E. degree in traffic information engineering from Xidian University, Xi'an, China, in 2021. She is currently pursuing the Ph.D. degree in cyberspace security with Air Force Engineering University, Xi'an. Her current research interests include vehicular communication networks, cloud computing for vehicular fault diagnosis, and communication countermeasures.

HUA XU received the B.S. and M.S. degrees in communication engineering from Air Force Engineering University, Xi'an, China, in 2001, and the Ph.D. degree in communication signal processing from Information Engineering University, Zhengzhou, China, in 2005. He is currently a Professor with Air Force Engineering University. His research interests include communication signal processing, blind signal processing, and communication countermeasures.

LINA ZHU (Member, IEEE) received the B.E. degree from the Suzhou University of Science and Technology, Suzhou, China, in 2009, and the Ph.D. degree in communication and information system from Xidian University, Xi'an, China, in 2015. She is currently a Lecturer with the State Key Laboratory of Integrated Services Networks, Xidian University. Her current research interests include mobility model, trust forwarding, routing, and MAC protocols in vehicular networks.