A Parallel Intelligence-Driven Resource Scheduling Scheme for Digital Twins-Based Intelligent Vehicular Systems

Junchao Yang, Feng Lin, Chinmay Chakraborty, Keping Yu, Zhiwei Guo, Tran Anh-Tu Nguyen, Joel Rodrigues

A Parallel Intelligence-Driven Resource Scheduling Scheme for Digital Twins-Based Intelligent Vehicular Systems

Article *in* IEEE Transactions on Intelligent Vehicles · April 2023

DOI: 10.1109/TIV.2023.3237960

CITATIONS
11

READS
93

7 authors, including:

Chinmay Chakraborty
Birla Institute of Technology, Mesra
324 PUBLICATIONS **4,265** CITATIONS
SEE PROFILE

Zhiwei Guo
Chongqing Technology and Business University
65 PUBLICATIONS **1,533** CITATIONS
SEE PROFILE

Anh-Tu Nguyen
Université Polytechnique Hauts-de-France
140 PUBLICATIONS **2,084** CITATIONS
SEE PROFILE

Joel Rodrigues
Senac Faculty of Ceará
1,274 PUBLICATIONS **36,144** CITATIONS
SEE PROFILE

A Parallel Intelligence-driven Resource Scheduling Scheme for Digital Twins-based Intelligent Vehicular Systems

Junchao Yang, Feng Lin, Chinmay Chakraborty, *Senior Member, IEEE,* Keping Yu, *Member, IEEE,* Zhiwei Guo, *Member, IEEE,* Anh-Tu Nguyen, *Member, IEEE,* and Joel J. P. C. Rodrigues, *Fellow, IEEE*

***Abstract*—Real-time digital twin technology can enhance traffic safety of intelligent vehicular system and provide scientific strategies for intelligent traffic management. At the same time, real-time digital twin depends on strong computation from vehicle side to cloud side. Aiming at the problem of delay caused by the dual dependency of timing and data between computation tasks, and the problem of unbalanced load of mobile edge computing servers, a parallel intelligence-driven resource scheduling scheme for computation tasks with dual dependencies of timing and data in the intelligent vehicular systems (IVS) is proposed. First, the delay and energy consumption models of each computing platform are formulated by considering the dual dependence of sub-tasks. Then, based on the bidding idea of the auction algorithm, the allocation model of computing resources and communication resources is defined, and the load balance model of the mobile edge computing (MEC) server cluster is formulated according to the load status of each MEC server. Secondly, joint optimization problem for offloading, resource allocation, and load balance is formulated. Finally, an adaptive particle swarm with genetic algorithm is proposed to solve the optimization problem. The simulation results show that the proposed scheme can reduce the total cost of the system while satisfying the maximum tolerable delay, and effectively improve the load balance of the edge server cluster.**

***Index Terms*—Parallel intelligence, digital twins, intelligent vehicular networks, resource scheduling, computation offloading.**

This work was supported in part by the Chongqing Municipal projects under grant KJCX2020035, CSTB2022BSXM-JCX0117 and cstc2020jcyj-msxmX0339, in part by National Natural Science Foundation of China under grant 62106029, and in part by Chongqing Technology and Business University projects under grant 2156004 and 212017. (Corresponding author: Chinmay Chakraborty)

Junchao Yang and Zhiwei Guo are with Chongqing Key Laboratory of Intelligent Perception and BlockChain Technology, National Research Base of Intelligent Manufacturing Services, Chongqing Technology and Business University, Chongqing 400067, China (e-mail: yangjc@ctbu.edu.cn; zwguo@ctbu.edu.cn).

Feng Lin is with School of automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: linfeng@cqupt.edu.cn).

Chinmay Chakraborty is with Birla Institute of Technology, Mesra, Jharkhand, India (email: cchakrabarty@bitmesra.ac.in).

Keping Yu is with School of Computer and Information Engineering, Bengbu University, Bengbu 233000, China, also with Graduate School of Science and Engineering, Hosei University, Tokyo 184-8584, Japan, and also with RIKEN Center for Advanced Intelligence Project, RIKEN, Tokyo 103-0027, Japan (e-mail: keping.yu@ieee.org).

Anh-Tu Nguyen is with INSA Hauts-de-France, Université Polytechnique Hauts-de-France, Valenciennes F-59313, France (email: tnguyen@uphf.fr).

Joel J. P. C. Rodrigues is with is with College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266555, China, and Instituto de Telecomunicações, 6201-001 Covilhã, Portugal (e-mail: joeljr@ieee.org).

I. Introduction

THE growing number of automobiles raised tremendous pressure on the transportation system in recent decades. Issues like traffic accidents, environmental pollution and energy wastage due to inadequate gasoline combustion due to traffic congestion need to be solved urgently [1]. Internet of Vehicles(IoV) is a key technology to realize Intelligent Traffic System (ITS) based on in-vehicle network, inter-vehicle network and mobile Internet [2], [3]. The continuous development of IoV in recent years and the commercialization of 5G networks have led to the emergence of IoV services, such as image-assisted navigation, augmented reality driving, identity recognition, natural language processing, image/audio/video processing and other computation intensive and latency-sensitive applications. Usually these applications require a large amount of computation resources, storage resources and energy consumption, but the computation capacity of the vehicle's terminal device is difficult to meet the maximum tolerable latency of those applications [4], as well as the energy consumption of the vehicle is limited. This will affect the quality of experience (Quality of Experience, QoE) of vehicle users for in-vehicle applications, as well as the safety of vehicle driving [5]. Therefore, such computation-intensive and latency-sensitive tasks pose a great challenge to the development of IoV [6].

Computation tasks in IoV can be classified into latency-sensitive and non-latency-sensitive tasks based on their latency characteristics [7]–[9]. For latency-sensitive tasks, such as autonomous driving, AR/VR applications, and online games with high latency requirements, while such applications are related to driving safety and require a large amount of computation power, while the vehicle itself has limited computation power and energy. Therefore, computation offloading is required to meet the maximum tolerable latency of the tasks. The non-latency-sensitive applications, such as music and video downloads do not have high requirements of latency, and exceeding a certain latency does not have a significant impact on the user experience. Computation tasks can be divided into decomposable and indecomposable computation tasks according to their computation characteristics [10]. Most of the computation tasks are decomposable tasks, so the computation tasks can be split into multiple sub-tasks for distributed computation offloading, which can be computed in parallel on multiple computing platforms and also reduce

the computation pressure of computing platform. However, after the task is divided into multiple sub-tasks, there is a dependency relationship between the sub-tasks, i.e the result of the former sub-task is the input of the latter one, so the latter sub-task must wait for the completion of the former sub-task, which will also bring additional latency of the computation task [11].

In order to cope with the limitation of computing power of vehicles, Mobile Cloud Computing (MCC) technology was introduced into the IoV, MCC extends the computing power of cloud computing to vehicles and uses cloud computing applications to process and serve computation intensive applications [12]. This MCC solution is to transfer all or some of the computation intensive tasks that cannot be satisfied by the computation power of the vehicles to a remote cloud server over a cellular network, and then feedback the computation results to the vehicle [13]. For example, in the work [14], [15], a cloud migration decision algorithm for computation intensive tasks is proposed to overcome the problem of limited computation resources of mobile devices, which determines whether to compute at the remote cloud server or at the mobile device based on the size of the task, and the results show that the computation latency of computation intensive tasks can be effectively reduced by utilizing the MCC technology. Although mobile cloud computing can provide powerful computing power for vehicles and solve the problem of insufficient computing resources for vehicles, the remote cloud servers are usually far away from vehicles because of the distance [16], [17]. Moreover, for computation tasks with relatively large amount of data size, such as autonomous driving, image and speech processing applications, it may lead to high transmission latency, which is unacceptable for those applications [18]. Secondly, a large number of vehicles offload computation tasks to remote cloud computing centers, which can put a huge pressure on the backbone network and may lead to network blockage or even paralysis [11],hence, MCC is not an ideal solution for computation task offloading in IoV.

In order to further reduce the transmission latency of computation tasks, mobile edge computing (MEC) is introduced into the vehicular network [19]–[21]. MEC mainly deploys servers at the edge of road, such as on Road Side Unit (RSU), Base Station (BS), etc. Thus, computation resources and storage resources are pushed to the edge of the road so that computation services can be provided near the vehicles, thus reducing the transmission latency of computation tasks and the return latency of computation results [22]. Compared with MCC, MEC not only reduces the transmission latency of computation tasks and the pressure on the backbone network, but also has the feature of flexible deployment, but the computation capacity of MEC is still insufficient to cope when there are too many vehicles needs to offload computation tasks. Therefore, the work [23] proposed MCC cooperated with MEC for offloading computation tasks, by offloading part of the computation tasks to the MEC server for computation through V2I, and by offloading the computation tasks to the MCC server through V2N, part of the computation can also be performed at the vehicle, thus effectively overcoming the problems of large transmission and return latencys and network blockage caused by offloading to the MCC, and avoiding the problem of insufficient computation resources at the MEC when too many task of are offloaded.

In work [24], the computation task is offloaded to the adjacent vehicles to assist the computation, and the dependent sub-task is modeled as a task scheduling problem. With the goal of minimizing the latency, Min-min, Max-min and HEFT algorithms are used to solve the task scheduling problem. The work [25] considered the situation of single MEC server with multiple users, offloaded the task to MEC or local computation, and proposed a binary search method to obtain the optimal offloading strategy with the goal of minimizing the weighted sum of latency and energy consumption. The work [26] presented a computation offloading strategy based on Particle Swarm Optimization (PSO). The computation task is offloaded to the local and MEC for computation. The latency and energy consumption are balanced, and the load balance of MEC server is considered. The work [27] proposed a joint computation offloading strategy and the optimization objective was to minimize the latency. The work [28] used the joint offloading method, and proposed a model-free method based on reinforcement learning to minimize the latency under the constraint of energy consumption and task dependency. The objective of work [29]is to minimize the weighted sum of latency and energy consumption. The computation task is offloaded to the local, MEC server and remote cloud for computation. The task-dependent migration scheme is considered and an improved Genetic Algorithm (GA) is used to solve the offloading decision [30].

In the aforementioned research, few studies jointly considered offloading strategy, resource allocation, load balance of MEC server cluster in the optimization objective, which leads the computing resources have not been fully utilized [12], [31]–[33]. In addition, the timing and data dual dependent tasks in IoV is not well studied [18]. Motivated by this, this paper proposes a joint computation offloading and resource allocation algorithm based on dual dependent tasks. This algorithm first considers the dependent tasks of multiple vehicles offloading to multiple computing platforms (i.e, local vehicle, MEC server , idle vehicle,cloud server) at the same time, and the latency and energy consumption models of each computing platform is formulated. Then, based on the bidding idea of auction algorithm, the allocation model of computing resources and communication resources is defined. According to the load state MEC server, the load balance model of MEC server cluster is formulated. Secondly, the optimization objective of offloading, resource allocation and load balance of MEC server is jointly constructed as Mixed-Integer Nonlinear Programming (MINLP) problem. Finally, combined with the characteristics of fast convergence of particle swarm optimization and strong global search of genetic algorithm and an adaptive hybrid particle swarm with genetic algorithm is proposed to solve the optimization problem.

II. Intelligent Vehicular Systems Model

A. Network Model

The intelligent vehicular system model of this paper is shown in Fig.1. In this paper, we consider deploying the

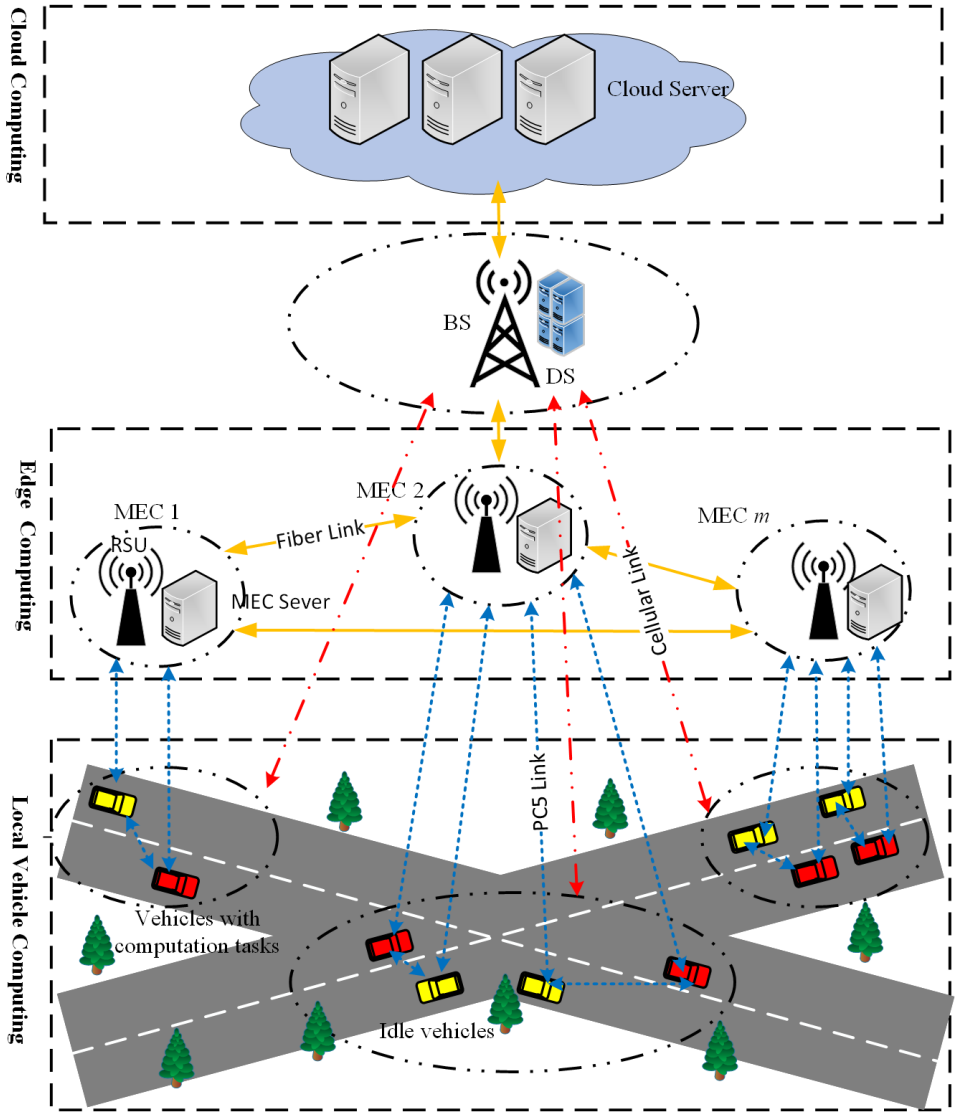

Fig. 1: Intelligent Vehicular Systems Model

MEC server on the RSU, without losing generality, and define the MEC server set that provides the computation offloading service within the coverage of BS as M = $\{mec_1, mec_2, ..., mec_n, ..., mec_m\}$, The vehicle set that needs to be offloaded for the computation task is V = $\{v_1, v_2, ..., v_i, ..., v_k\}$, and each vehicle has a computation intensive task S_i,The set of computation intensive tasks is $S = \{S_1, S_2, ..., S_i, ..., S_k\}$ Where S_i is a separable task, including several sub-tasks with temporal and data dependence. Assuming that each offloaded vehicle has a corresponding idle vehicle to provide computation offload, each MEC server mec_n has a separate computing resource F_n and storage resource W_n. In this paper, Dispatch Server (DS) is set up at BS. Its function is to make a unified offloading strategy for all offloading vehicles covered by BS, and to manage and coordinate the resource allocation of all MEC servers within its scope, as well as the bandwidth resource allocation of BS. The decision-making process of computation offloading is regarded as semi-dynamic. Within a scheduling cycle T(namely several hundred milliseconds), the vehicle set V and the computation intensive task set S remain unchanged, however V and S may change in different scheduling cycles. Optical fiber links are adopted between RSU and RSU, RSU and BS, BS and cloud servers, respectively. PC5 communication is used between vehicles, vehicles and RSU, respectively. And cellular network communication is used between vehicles and BS.

In order to avoid the interference between cellular links and PC5 links, this paper assumes that the system allocates different bandwidth resources to cellular links and PC5 links. The total bandwidth available for BS is B_1, and each PC5 link can be allocated to an orthogonal channel with bandwidth B_2. The uplink and downlink of the vehicle are Rayleigh channels.

The uplink / downlink data transmission rates of vehicle v_i and BS are defined as R_i^1 and R_b^i, respectively :

$$R_i^1 = \lambda_i B_1 \log_2 \left(1 + \frac{P_i^u h_1}{N_1}\right) \quad (1)$$

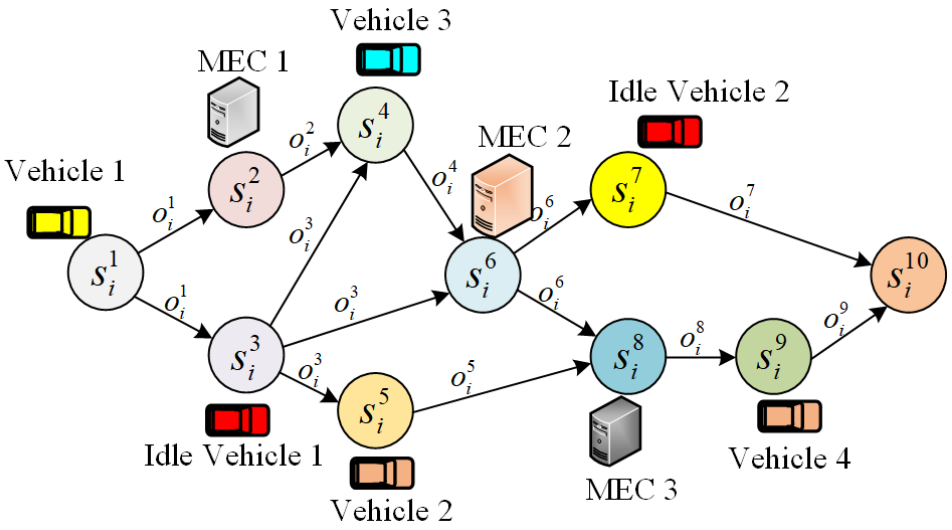

Fig. 2: Dependency Model of computation tasks

$$R_b^i = \lambda_i B_1 \log_2\left(1 + \frac{P_b^u h_1}{N_1}\right) \tag{2}$$

where λ_i represents the BS bandwidth allocation factor of task S_i of vehicle v_i, P_i^u represents the transmitting power of vehicle v_i, P_b^u represents the transmitting power of BS, h_1 represents the channel gain of the transmission link between vehicle v_i and BS, and N_1 represents the Gaussian white noise power of the transmission link between vehicle and BS.

Vehicle v_i and MEC, idle vehicle uplink data transmission rate R_i^2:

$$R_i^2 = B_2 \log_2\left(1 + \frac{P_i^u h_2}{N_2}\right) \tag{3}$$

The downlink data transmission rates of v_i, MEC and idle vehicles are R_m^i, R_a^i:

$$R_m^i = B_2 \log_2\left(1 + \frac{P_m^u h_2}{N_2}\right) \tag{4}$$

$$R_a^i = B_2 \log_2\left(1 + \frac{P_a^u h_2}{N_2}\right) \tag{5}$$

where P_m^u and P_a^u represent the transmission power of MEC server and idle assistance vehicle respectively; h_2 represents the channel gain of transmission link between vehicle and MEC and idle assistance vehicle; N_2 represents the Gaussian white noise power of transmission link between vehicle and MEC and idle assistance vehicle.

This paper considers the computation tasks can be processed in the local vehicle, or offloaded to MEC server, idle vehicle and cloud server for computation, and then the computation results are returned. The offloading of cloud server needs to be transmitted to BS through cellular network, and then to cloud server through optical fiber. Therefore, it results large communication latency of task transmission. The communication latency of MEC server offloading is relatively low, however the computation ability is limited. The characteristics of idle vehicle offloading are low communication latency and weak computing ability. Therefore, according to the characteristics of each computing platform, as well as the timing and data dependence of sub-tasks, a reasonable offloading strategy can effectively reduce the total system cost.

B. Task Dependent model

In this paper, the computation-intensive task set S of multiple vehicles is mainly studied. The computation-intensive task S_i of offloading vehicle V_i is represented as $S_i = \{i, T_i^{\max}, I_i, Sm_i, J\}$, where i represents the number of computation task S_i of vehicle V_i, T_i^{max} represents the maximum tolerance latency of task S_i, and I_i represents the amount of data of task S_i. Sm_i represents the number of sub-task $Sm_i = \left\{s_i^1, s_i^2, s_i^3 ...., s_i^j, ..., s_i^J\right\}$ that can be independently calculated, and J represents the number of sub-tasks that task S_i is segmented . sub-tasks have dual dependence on timing and data, sub-tasks rely on the execution results of other sub-tasks, and then can successfully perform this task. As shown in Fig.2, the dependency between sub-tasks is represented by Directed Acyclic Graph (DAG), where the nodes of DAG represent the sub-tasks that can be independently calculated, each edge of the graph represents the dependency between sub-tasks, and the weight represents the amount of data transmitted between sub-tasks.

In this paper, each sub-task of the offloading computation task S_i of vehicle V_i is modeled as a ten-tuple $s_i^j = \{j, \mathrm{I}_i^j, \mathrm{O}_i^j, \mathrm{U}_i^j, E_i^j, \mathrm{Z}_i^j, L_i^j, Y_i^j, K_i^j, Q_i^j\}$. where j represents the number of sub-tasks, and I_i^j represents the data volume of sub-task s_i^j. $\mathrm{O}_i^j = \delta \cdot I_i^j$ is the output result data volume, and δ is the output data volume coefficient, indicating the relationship between the output data volume and the input data volume. $\mathrm{U}_i^j \in \{0, 1, 2, 3\}$ represents the location of the sub-task s_i^j to be offloaded, which represents the local computation, offloaded to the MEC server, to the idle vehicle and the remote cloud server, respectively. Z_i^j represents the

number of the MEC server corresponding to the U_i^j=1, in other cases Z_i^j=0. L_i^j denotes the transmission latency of sub-task s_i^j from offloading vehicle v_i to computing position U_i^j, and Y_i^j denotes the sum of the computation latency and the computation result O_i^j of sub-task s_i^j at computing position U_i^j to the next computing position K_i^j. K_i^j denotes the set of computing positions of follow-up sub-tasks of s_i^j. For example, the follow-up sub-tasks of s_i^3 in Fig.2 are s_i^4, s_i^5 and s_i^6, so K_i^3=$\{U_i^4, U_i^5, U_i^6\}$, Q_i^3=$\{Z_i^4, Z_i^5, Z_i^6\}$ and Q_i^j represent the index of MEC server when the follow-up sub-task is offloaded to MEC. E_i^j represents the total energy consumption of sub-task s_i^j from offloading vehicle v_i to calculating result O_i^j within the computation position set K_i^j. For example, the dependent task in Fig.2 ends with sub-task s_i^{10}, so the K_i^{10}=$\{0\}$ of s_i^{10}, that is, the result of the computation task is finally returned to the vehicle V_i.

III. Computation Model Based on Dual Dependent Tasks

As shown in Fig. 3, the relationship between the offloading of the computation sub-task s_i^j and the transmission of the computation result O_i^j between the local vehicle, the idle assistance vehicle, the MEC server and the remote cloud server is shown. For example, when vehicle v_i offload s_i^j to MEC server, it is divided into two cases : the local MEC server where the vehicle is associated and other MEC servers under DS, so as to consider whether there is a transmission latency between MEC servers. Offloading s_i^j to the cloud server requires transport through BS and then transmitting to the cloud server. Similarly, it is also necessary to consider whether there is a transmission latency between the MEC servers and the transmission latency between the MEC server and the BS when the computation result O_i^j is returned to the computation position of the follow-up sub-task. And in the following we will give the latency and energy consumption model under different situations.

A. Local vehicle computing model

When $U_i^j = 0$, it means that sub-task s_i^j is executed in local vehicle v_i, and vehicle v_i is represented by a five-tuple V_i= $\{d_i, G_v, F_i, P_i, P_i^u\}$. where d_i represents the number of MEC server where vehicle v_i is located. G_v is the number of CPU cycles required for the local computation of 1 bit data for vehicles, i.e, cycle / bit. F_i is the local computing power of vehicle v_i, that is, the number of CPU cycles per second. P_i is the device power when the task is executed locally in vehicle V. If sub-task s_i^j is locally executed, and transmission latency $L_i^j = 0$. The computation latency of sub-task s_i^j offloading to local is t_{ij}^{lc}:

$$t_{ij}^{lc} = \frac{\mathrm{I}_i^j G_v}{F_i} \tag{6}$$

In this study, it is considered that the vehicle can communicate with idle vehicles, MEC servers and BSs at the same time.

$$t_{ij}^{\text{Omrelay}} = \min\left(\left|d_i - Z_i^j\right|, 1\right) \cdot \frac{O_i^j}{R_f^m} \tag{7}$$

where $t_{ij}^{Omrelay}$ denotes the transmission latency of the calculation result O_i^j of sub-task s_i^j over the fiber link between MEC servers, and R_f^m represents the data transmission rate of the optical fiber link between the MEC servers.

$$t_{ij}^{\text{Ocrelay}} = \frac{O_i^j}{R_f^c} \tag{8}$$

where $t_{ij}^{Ocrelay}$ is the transmission latency of the calculation result O_i^j of sub-task s_i^j over the fiber link between BS and the cloud server, and R_f^c is the transmission rate of the optical fiber link between BS and cloud server.

In this paper, we define the following function to model the latency and energy consumption during the task offloading. When k_i^j=U_i^j, ξ=0, it shows that the computation results need to be back-transmitted is the current position, so there is no back-transmitting computation results and no back-transmitting latency. When $k_i^j \neq U_i^j$, ξ= 1, the computation results of sub-task s_i^j need to be transmitted to the next computation position, so there is a return latency.

$$\xi = \min\left(|k_i^j - U_i^j|, 1\right) \tag{9}$$

Since the data transmission with idle vehicles may require the relay of other vehicles, the relay latency between vehicles and idle vehicles is t_{aw}. $\delta(t)$ is the impulse function, Under the condition of $k_i^j \neq U_i^j$, such as when k_i^j= 1, $\min(\delta(k_i^j-1), 1) = \min(\infty, 1) = 1$; when $k_i^j \neq 1$, $\min(\delta(k_i^j-1), 1) = \min(0, 1) = 0$, indicating that the computation position of the follow-up sub-task is MEC server, only the computation result O_i^j is transmitted to the MEC server latency, the other latency is 0.

$$\Gamma_1 = \min(\delta(k_i^j-1), 1) \tag{10}$$

$$\Gamma_2 = \min(\delta(k_i^j-2), 1) \tag{11}$$

$$\Gamma_3 = \min(\delta(k_i^j-3), 1) \tag{12}$$

So the computation result O_i^j of the sub-task s_i^j is transmitted from the local vehicle to the latency of the computation position set K_i^j of the rear drive sub-task, and the maximum latency should be taken, that is, t_{ij}^{lr}:

$$t_{ij}^{lr} = \max_{k_i^j \in K_i^j} \left\{ \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_i^2} + t_{ij}^{Omrelay}\right) + \Gamma_2 \cdot \left(\frac{O_i^j}{R_i^2} + t_{aw}\right) \\ + \Gamma_3 \cdot \left(\frac{O_i^j}{R_i^1} + t_{ij}^{Ocrelay}\right) \end{array} \right] \right\} \tag{13}$$

When $d_i = Z_i^j$, the MEC server that represents the vehicle v_i to unload is the current MEC server. Therefore, $|d_i - Z_i^j| = 0$, $\min\left(|d_i - Z_i^j|, 1\right) = 0$, so there is no transmission latency between MEC servers, so $t_{ij}^{Omrelay} = 0$. When $d_i \neq Z_i^j$, it indicates that the MEC server to be

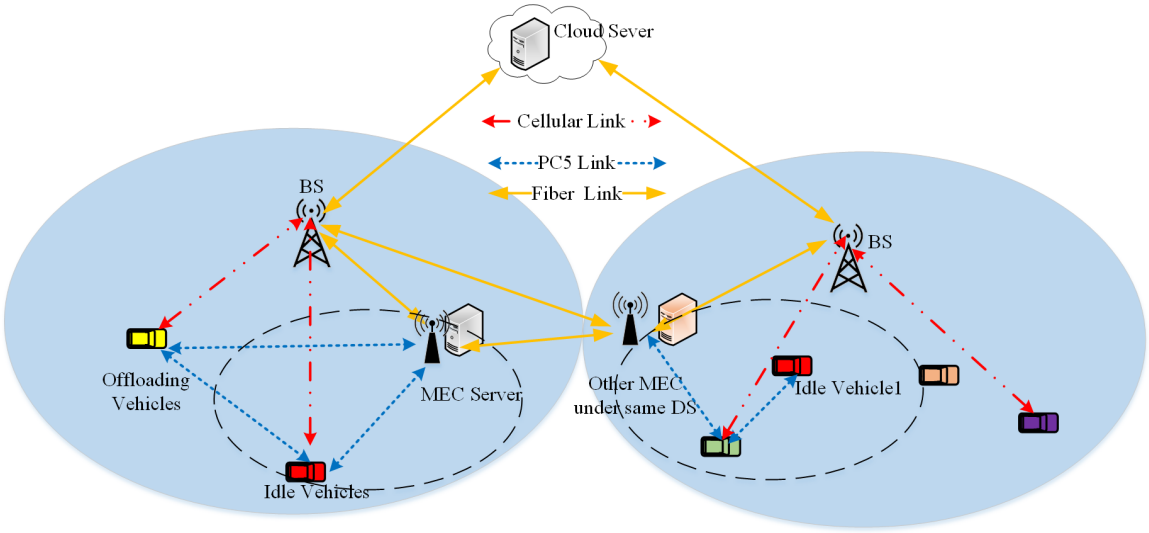

Fig. 3: Computation tasks offloading scenarios in IVS

offloaded by vehicle v_i is not the current MEC server. $|d_i - Z_i^j| \geq 1$, there is the transmission latency between MEC servers, $\min\left(|d_i - Z_i^j|, 1\right) = 1$, then the transmission latency $t_{ij}^{Omrelay} = \frac{O_i^j}{R_f^m}$ between MEC servers. The computation latency of sub-task s_i^j in local computation and the latency sum of computation result O_i^j transmitted to the computation position set K_i^j of follow-up sub-task is $t_{ij}^{lc} + t_{ij}^{lr}$.

$$e_{ij}^{lr} = \sum_{k_i^j \in K_i^j} \left\{ \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_i^2} \cdot P_i^u + t_{ij}^{Omrelay} \cdot P_m^u \right) \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_i^2} + t_{aw} \right) \cdot P_i^u \\ +\Gamma_3 \cdot \left(\frac{O_i^j}{R_i^1} \cdot P_i^u + t_{ij}^{Ocrelay} \cdot P_b^u \right) \end{array} \right] \right\} \tag{14}$$

where e_{ij}^{lr} represents the energy consumption of the computation result from the local vehicle to the computation position of all follow-up sub-tasks.

The total energy consumption of sub-task s_i^j offloaded to local computations e_{ij}^{local} if formulated as:

$$e_{ij}^{local} = t_{ij}^{lc} \cdot P_i + e_{ij}^{lr} \tag{15}$$

B. MEC server computation model

When $U_i^j = 1$, and $Z_i^j = n$, the sub-task s_i^j is performed on the mec_n server, which is represented by a six-tuple $ME_n = \{n, G_m, F_n, W_n, P_m, P_m^u\}$. where n represents the number of MEC server, G_m is the number of CPU cycles required by MEC server to process 1 bit data, unit cycle / bit. F_n is the computing resource cycle / s for the mec_n server. P_m is the device power when the task is executed on the MEC server, and P_m^u is the transmit power of the MEC server.

Transmission latency of sub-task s_i^j from offloading vehicle v_i to MEC server t_{ij}^{ms}:

$$\begin{aligned} t_{ij}^{ms} &= t_{ij}^{vm} + t_{ij}^{Imrelay} \\ &= \frac{I_i^j}{R_i^2} + \min\left(\left|d_i - Z_i^j\right|, 1\right) \cdot \frac{I_i^j}{R_f^m} \end{aligned} \tag{16}$$

where t_{ij}^{vm} represents the latency of sub-task s_i^j transmitting to the MEC server where the vehicle is located, and $t_{ij}^{Imrelay}$ represents the transmission latency of the optical fiber link between the sub-task s_i^j and the MEC server.

The computation latency of sub-task s_i^j on MEC server t_{ij}^{mc}:

$$t_{ij}^{mc} = \frac{I_i^j \cdot G_m}{F_n^i} \tag{17}$$

where F_n^i denotes the computing resource allocated to vehicle v_i by MEC server, unit cycle / s. The latency of the computation result O_i^j of the sub-task s_i^j from the MEC server to the computation position set K_i^j of the follow-up sub-task is t_{ij}^{mr}:

$$t_{ij}^{mr} = \max_{k_i^j \in K_i^j, q_i^j \in Q_i^j} \left\{ \begin{array}{l} \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_m^i} + t_{ij}^{Omrelay} \right) \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_m^i} + t_{ij}^{Omrelay} \right) \\ +\Gamma_3 \cdot \left(t_{ij}^{Obrelay} + \frac{O_i^j}{R_f^c} \right) \end{array} \right] \\ +\Theta \cdot t_{ij}^{Omrelay} \end{array} \right\} \tag{18}$$

$$t_{ij}^{Obrelay} = \frac{O_i^j}{R_f^b} \tag{19}$$

$$\Theta = \min\left(|\delta(k_i^j - 1) \cdot (q_i^j - d_i)|, 1\right) \tag{20}$$

Since the idle assistance vehicle is connected to the MEC server through the PC5 link, the return latency when the computation result O_i^j is transmitted from the MEC server to the idle vehicle is $\frac{O_i^j}{R_m^i} + t_{ij}^{Omrelay}$. $t_{ij}^{Obrelay}$ represents the

latency of the computation result O_i^j of the sub-task s_i^j from the MEC server through the optical fiber to the BS, and R_f^b is the transmission rate of the optical fiber link between the MEC server and the BS. $\Theta \cdot t_{ij}^{Omrelay}$ indicates that when the current computing location is MEC server, it may not be the same MEC server, and may have a transmission latency between MEC servers.

The sum of computation latency and computation result O_i^j of sub-task s_i^j on MEC server to follow-up sub-task computation position set s_i^j is written as: $t_{ij}^{mc} + t_{ij}^{mr}$.

The energy consumption of the computation results transmitted from the current MEC server to all the position of follow-up sub-tasks e_{ij}^{mr} is defined.

$$e_{ij}^{mr} = \sum_{k_i^j \in K_i^j} \left\{ \begin{array}{l} \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_m^i} + t_{ij}^{Omrelay} \right) \cdot P_m^u \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_m^i} + t_{ij}^{Omrelay} \right) \cdot P_m^u \\ +\Gamma_3 \cdot \left(t_{ij}^{Obrelay} \cdot P_m^u + \frac{O_i^j}{R_f^c} \cdot P_b^u \right) \end{array} \right] \\ +\Theta \cdot t_{ij}^{Omrelay} \cdot P_m^u \end{array} \right\} \tag{21}$$

The total energy consumption of sub-task s_i^j offloading to MEC e_{ij}^{mec} can be defined as:

$$e_{ij}^{mec} = t_{ij}^{vm} \cdot P_i^u + t_{ij}^{Imreleay} \cdot P_m^u + t_{ij}^{mc} \cdot P_m + e_{ij}^{mr} \tag{22}$$

C. Computation model of idle vehicle

When $U_i^j = 2$, the sub-task s_i^j performs in idle assistance vehicles, which are represented by a quad $A = \{G_a, F_a, P_a, P_a^u\}$. where G_a is the number of CPU cycles, unit cycles / bit, required to idle vehicles in processing 1 bit data. F_a is idle vehicle computing capacity, unit cycle / s. P_a is the equipment power when the task is idle to assist the vehicle, and P_a^u is the transmitting power of the idle to assist the vehicle.

sub-task s_i^j from offloading vehicle v_i to idle vehicle transmission latency t_{ij}^{as}:

$$t_{ij}^{as} = t_{ij}^{va} + t_{aw} = \frac{I_i^j}{R_i^2} + t_{aw} \tag{23}$$

where t_{ij}^{va} represents the latency of data transmission from sub-task s_i^j to idle assistance vehicles, and t_{aw} is the relay latency between vehicles and idle vehicles.

The computation latency of sub-task s_i^j in idle vehicle t_{ij}^{ac}:

$$t_{ij}^{ac} = \frac{I_i^j \cdot G_a}{F_a} \tag{24}$$

The result O_i^j of the sub-task s_i^j is transmitted from the idle vehicle to the follow-up sub-task computation location set K_i^j with time latency t_{ij}^{ar}:

$$t_{ij}^{ar} = \max_{k_i^j \in K_i^j} \left\{ \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_a^i} + t_{aw} \right) \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_i^2} + t_{ij}^{Omrelay} \right) \\ +\Gamma_3 \cdot \left(\frac{O_i^j}{R_i^1} + t_{ij}^{Ocrelay} \right) \end{array} \right] \right\} \tag{25}$$

No.	Node Sequence						
$Path_i^1$	1	2	4	6	7	10	
$Path_i^2$	1	2	4	6	8	9	10
$Path_i^3$	1	3	4	6	7	10	
$Path_i^4$	1	3	4	6	8	9	10
$Path_i^5$	1	3	6	7	10		
$Path_i^6$	1	3	6	8	9	10	
$Path_i^7$	1	3	5	8	9	10	

Fig. 4: An example of critical paths of computation tasks

The computation latency of sub-task s_i^j in idle assistance vehicle and the latency sum of the computation result O_i^j returning to the computation position set K_i^j of follow-up sub-task is $t_{ij}^{ac} + t_{ij}^{ar}$

The energy consumption of the computation process from idle vehicles to all follow-up sub-task locations e_{ij}^{ar} is defined as:

$$e_{ij}^{ar} = \sum_{k_i^j \in K_i^j} \left\{ \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_a^i} + t_{aw} \right) \cdot P_a^u \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_i^2} \cdot P_a^u + t_{ij}^{Omrelay} \cdot P_m^u \right) \\ +\Gamma_3 \cdot \left(\frac{O_i^j}{R_i^1} \cdot P_a^u + t_{ij}^{Ocrelay} \cdot P_b^u \right) \end{array} \right] \right\} \tag{26}$$

Total energy consumption of sub-task s_i^j offloaded to idle Vehicle e_i^{assit} is formulated as:

$$e_{ij}^{assit} = t_{ij}^{as} \cdot P_i^u + t_{ij}^{ac} \cdot P_a + e_{ij}^{ar} \tag{27}$$

D. Cloud Server Computing Model

When $U_i^j = 3$, sub-task s_i^j is performed on the cloud server, which is represented by a quad $C = \{G_c, F_c, P_c, P_c^u\}$. where G_c is the number of CPU cycles required by the cloud server to process 1 bit data, unit cycle / bit. F_c is the computing power of cloud server, unit cycle / s. P_c is the device power when the task is executed on the cloud server, and P_c^u is the transmitting power of the cloud server.

The transmission latency of sub-task s_i^j from offloading vehicle v_i to cloud server t_i^{cs} is formulated as:

$$t_{ij}^{cs} = t_{ij}^{vb} + t_{ij}^{Icrelay} = \frac{I_i^j}{R_i^1} + \frac{I_i^j}{R_f^c} \tag{28}$$

where t_{ij}^{vb} represents the latency of data transmission from sub-task s_i^j to BS, $t_{ij}^{Icrelay}$ represents the transmission latency of optical fiber link between sub-task s_i^j and cloud server, and R_f^c is the transmission rate of optical fiber link between cloud server and BS.

The computing latency of sub-task s_i^j in cloud server is t_{ij}^{cc}:

$$t_{ij}^{cc} = \frac{I_i^j \cdot G_c}{F_c} \tag{29}$$

The computation result O_i^j of the sub-task s_i^j is transmitted from the cloud server to the latency t_{ij}^{cr} of the computation position set K_i^j of the follow-up task:

$$t_{ij}^{cr}= \max_{k_i^j \in K_i^j} \left\{ \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_f^c} + \frac{O_i^j}{R_b^i} \right) \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_f^c} + \frac{O_i^j}{R_f^b} \right) \\ +\Gamma_3 \cdot \left(\frac{O_i^j}{R_f^c} + \frac{O_i^j}{R_b^i} \right) \end{array} \right] \right\} \tag{30}$$

The sum of computing latency and computing result O_i^j of sub-task s_i^j in cloud server back to the computing position set K_i^j of follow-up task is $t_{ij}^{cc} + t_{ij}^{cr}$

The energy consumption of the computation process from idle vehicles to all follow-up sub-task locations e_{ij}^{cr} is defined as:

$$e_{ij}^{cr}= \sum_{k_i^j \in K_i^j} \left\{ \xi \cdot \left[\begin{array}{l} \Gamma_1 \cdot \left(\frac{O_i^j}{R_f^c} \cdot P_c^u + \frac{O_i^j}{R_b^i} \cdot P_b^u \right) \\ +\Gamma_2 \cdot \left(\frac{O_i^j}{R_f^c} \cdot P_c^u + \frac{O_i^j}{R_f^b} \cdot P_b^u \right) \\ +\Gamma_3 \cdot \left(\frac{O_i^j}{R_f^c} \cdot P_c^u + \frac{O_i^j}{R_b^i} \cdot P_b^u \right) \end{array} \right] \right\} \tag{31}$$

Total energy consumption of sub-task s_i^j offloading to cloud servers e_{ij}^{cloud} is formulated as:

$$e_{ij}^{cloud} = t_{ij}^{cs} \cdot P_i^u + t_{ij}^{cc} \cdot P_c + e_{ij}^{cr} \tag{32}$$

E. Total system cost formulation

Since we consider the dual dependence of timing and data between sub-tasks. In this study, the vehicle is assumed to be a multi-antenna offloading mode, so the total latency is not a simple sum of sub-task latency. Therefore, considering the total time latency of the task S_i, this paper uses the Critical Path Method (CPM), which is the longest path in this paper. All paths are obtained by the breadth-first traversal algorithm of DAG graph. For example, there are seven paths to complete the dependency task as shown in Fig.4.

Assuming that the path set of task S_i of vehicle v_i is $Path_i = \{path_i^1, path_i^2, ..., path_i^p\}$, there are a total of p paths.

The total latency of task S_i path $path_i^p$ of vehicle v_i is T_i^p :

$$\begin{aligned} T_i^p &= T_i^{send} + \sum_{j=1 \wedge j \in Path_i^p} Y_i^j \\ &= \max \left(\sum_{j=1, U_i^j=1}^{J} L_i^j, \sum_{j=1, U_i^j=2}^{J} L_i^j, \sum_{j=1, U_i^j=3}^{J} L_i^j \right) \\ &+ \sum_{j=1 \wedge j \in Path_i^p} Y_i^j \end{aligned} \tag{33}$$

Where T_i^{send} represents the maximum latency of simultaneous transmission task S_i completion. Since all tasks that are not calculated locally need to be offloaded, the latency in this part should be the maximum latency in the MEC, idle assistance vehicles and cloud servers. And the dependencies between tasks must wait until the precursor task is finished and the results are passed to the next sub-task to start execution, so the computation latency and the backhaul latency on the $path_i^p$ should be additive.

The total offloading latency of vehicle v_i task S_i is T_i, and the total energy consumption is E_i. The latency sum of all computation intensive task set S of vehicle set V, namely, the total system latency is T, and the total energy consumption is E:

$$T = \sum_{\mathrm{i}=1}^{\mathrm{k}} \max \left(T_i^1, T_i^2, ..., T_i^p \right) \tag{34}$$

$$E = \sum_{i=1}^{k} \sum_{j=1}^{J} E_i^j \tag{35}$$

The total system cost is H:

$$H = \beta T + (1 - \beta) E \tag{36}$$

Where $\beta \in (0, 1)$ is the latency coefficient, $(1 - \beta)$ is the energy consumption coefficient. β=0.8 in this paper.

IV. Resource Allocation and load balance Model for MEC Server

A. MEC Server Resource Allocation Model

The scheduling server calculates the amount of tasks that each vehicle offloads to each MEC server, and the amount of offloading of the cloud server. According to the bidding idea of auction algorithm, the computation resources and BS bandwidth are allocated to each offloading vehicle.Vehicle v_i with computation task S_i for MEC server computing resources bidding for Fb_n^i:

$$\begin{aligned} & F_n^i \\ =& Fb_n^i \cdot F_n \\ =& \left(a_1 \cdot \frac{\sum_{j=1 \wedge Z_i^j=n}^{J} I_i^j}{\sum_{i=1}^{k} \sum_{j=1 \wedge Z_i^j=n}^{J} I_i^j} + a_2 \cdot \min(\delta(d_i - n), 1) \right) \cdot F_n \end{aligned} \tag{37}$$

Where a_1=0.75, a_2=0.25. a_1 represents the weight of the computation resource request, and a_2 represents the priority weight of the vehicle. When d_i=n, $\min(\delta(d_i - n), 1) = \min(\infty, 1) = 1$; when $d_i \neq n$, $\min(\delta(d_i - n), 1) = \min(0, 1) = 0$. Because when vehicle v_i is under the mec_n server, the priority of vehicle v_i will be higher, which can reduce the return latency of computation results.

Vehicle v_i with computation task S_i bids for BS bandwidth resources, where BS bandwidth allocation factor λ_i:

$$B_i = \lambda_i \cdot B_1 = \frac{\sum_{j=1 \wedge U_i^j=3}^{J} I_i^j}{\sum_{i=1}^{k} \sum_{j=1 \wedge U_i^j=3}^{J} I_i^j} \cdot B_1 \tag{38}$$

Because the more vehicles offloaded to the cloud server, the more bandwidth resources should be allocated, which can reduce the transmission latency and the result backhaul latency.

The allocation of storage resources for mec_n server by task S of vehicle v_i is W_n^i:

$$W_n^i = \sum_{j=0 \wedge Z_i^j = n}^{J} \left(I_i^j + O_i^j \right) \tag{39}$$

Because all sub-tasks of task S_i offloaded to the mec_n server must be stored first, and the computation results of sub-tasks also need to be stored to be transmitted to the follow-up sub-task.

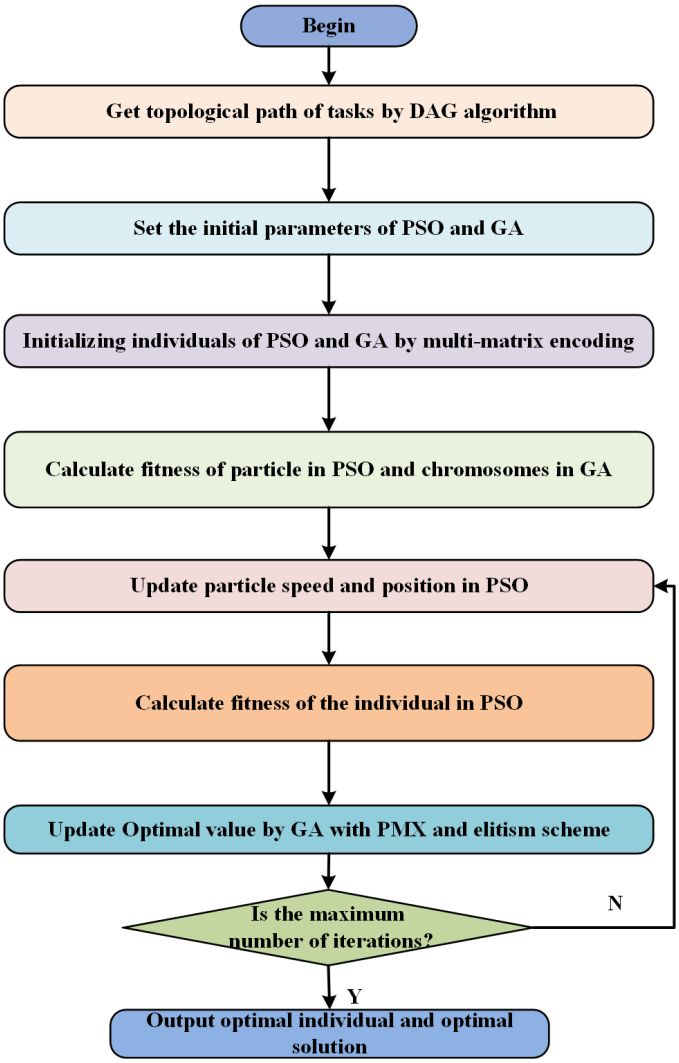

Fig. 5: The flow chart of proposed algorithm

B. MEC Server Load Balance Model

In the vehicle networking environment, due to the mobility of vehicles, the distribution of offloading vehicles under the MEC server is uneven, resulting in unbalanced load of the MEC server. It not only makes the resources not fully utilized, but also leads to poor QoE. Therefore, it is necessary to consider load balance of MEC servers when offloading strategies and resource allocation are made.

When balancing the load of the MEC server, it is first necessary to determine the load of the mec_n server. In this paper, the computing resources F_n and storage resources F_n of the MEC server are needed. Therefore, the determination of the load of the MEC server in this paper is mainly considered from two aspects : computing resources and storage resources.

The load state of mec_n server is expressed as $\gamma_n = (\gamma_n^{cpu}, \gamma_n^{mem})$, where γ_n^{cpu} represents the current computing resource usage of mec_n server, and γ_n^{mem} represents the current memory usage of mec_n server.

mec_n server CPU usage is G_n^{cpu}, memory usage is G_n^{mem}:

$$G_n^{cpu} = \frac{\gamma_n^{cpu}}{F_n}, \quad G_n^{mem} = \frac{\gamma_n^{mem}}{W_n} \tag{40}$$

Therefore, the load of mec_n server is G_n:

$$G_n = \alpha_1 \cdot G_n^{cpu} + \alpha_2 \cdot G_n^{mem} \tag{41}$$

where $\alpha_1 + \alpha_2 = 1$ and α_1 represent the degree of attention to the use of computing resources, and α_2 represents the degree of attention to the use of memory resources. This study pays the same attention to computing resources and storage resources, so $\alpha_1 = \alpha_2 = 0.5$.

Under the scheduling server, MEC server set is $M = \{mec_1, mec_2, ..., mec_n, ..., mec_m\}$, then the load set of MEC server cluster is $G_M = \{G_1, G_2, ..., G_n, ..., G_m\}$.

In this paper, the standard deviation ω of MEC server cluster load is used to represent the load balance of cluster:

$$\omega = \sqrt{\frac{1}{m} \cdot \sum_{n=1}^{m} \left(G_n - \frac{1}{m} \cdot \sum_{n=1}^{m} G_n \right)^2} \tag{42}$$

The smaller ω indicates the higher load balance of MEC server cluster, and the larger ω indicates the lower load balance of MEC server cluster.

C. Problem Formulation

In the scheduling cycle Δt, the whole process of offloading system is : Firstly, k offloading vehicles upload the relevant information of their computing-intensive tasks, such as the dependency matrix of sub-tasks, the size of sub-tasks, etc. After receiving the offloading request of the vehicle, DS considers multi-platform offloading based on the uploaded information. The resource status of m MEC servers in DS range is comprehensively analyzed to minimize the cost (weighted sum of latency and energy consumption) and load balance of the whole system. The offloading, MEC server resource allocation and BS bandwidth resource allocation are jointly optimized.

Finally, the offloading strategy results are returned to the offloading vehicles, BS and MEC. Each vehicle completes the computation of offloading according to the strategy results. BS and MEC allocate bandwidth and computation resources respectively. Under the premise of meeting the dependence of each vehicle's computation intensive task S_i, as well as the

maximum tolerance of latency and resource constraints, the goal is to minimize the total cost and load balance of the joint offloading system. Therefore, the task offloading and resource allocation of the system are modeled as:

$$\begin{aligned}
&\min_{K,U,Z,F,W,B} [\mu \cdot H + (1-\mu) \cdot \omega] \\
&s.t. C1: \sum_{j=1}^{J} I_i^j = I_i \\
&C2: T_i \le T_i^{\max}, \forall i \in [1,k] \\
&C3: 0 \le F_n^i \le F_n, 0 \le \sum_{i=1}^{k} F_n^i \le F_n, \forall i \in [1,k], \forall n \in [1,m] \\
&C4: 0 < \lambda_i < 1, 0 < \sum_{i=1}^{k} \lambda_i \cdot B_1 \le B_1, \forall i \in [1,k] \\
&C5: 0 \le \sum_{i=1}^{k} \left[\sum_{j=1, U_i^j=1 \wedge Z_i^j=n}^{J} (I_i^j + O_i^j) \right] \le W_n, \forall n \in [1,m] \\
&C6: U_i^j \in \{0,1,2,3\}, Z_i^j, Q_i^j \in \{1,2,...,m\}, K_i^j \in \{0,1,2,3\}, \\
&\forall i \in [1,k], \forall j \in [1,J]
\end{aligned} \tag{43}$$

Where μ represents the weight of the total cost of the system in the objective function, and $(1-\mu)$ represents the weight of the load balance in the objective function. $K = [K_1, K_2, ..., K_i, ..., K_k]$ is the task-dependent topological matrix of all vehicles, and the i element in K corresponds to the topological vector K_i of the computing task S_i of vehicle v_i. $U = [U_1, U_2, ..., U_i, ..., U_k]$ is the offloading matrix of all vehicle computation tasks. The ith element in $U_i = [U_i^1, U_i^2, ..., U_i^j, ..., U_i^J]$ corresponds to the offloading strategy vector U of the computation task S_i of vehicle v_i, that is, the offloading position corresponding to the K_i neutron task. $Z = [Z_1, Z_2, ..., Z_i, ..., Z_k]$ represents the number of the MEC server when the offloading location is MEC. $Z_i = [Z_i^1, Z_i^2, ..., Z_i^j, ..., Z_i^J]$, when $Z_i^j = 0$, indicates not offloading to the MEC server. $F = [F_1, F_2, ..., F_n, ..., F_m]$ and $W = [W_1, W_2, ..., W_n, ..., W_m]$ are the computation resource allocation matrix and storage resource allocation matrix for all MEC servers, respectively. The nth row in F corresponds to the computation resource allocation vector $F_n = [F_n^1, F_n^2, ..., F_n^i, ..., F_n^k]$ for the mec_n server, and the nth row in W corresponds to the storage resource allocation vector $W_n = [W_n^1, W_n^2, ..., W_n^i, ..., W_n^k]$ for the MEC server. B is the bandwidth resource allocation vector of BS, and the elements in the vector represent the bandwidth allocation coefficient $B = [\lambda_1, \lambda_2, ..., \lambda_i, ..., \lambda_k]^{\mathrm{T}}$ for each vehicle.

In the constraint condition, $C1$ denotes the amount of data that vehicle v_i cuts S_i into multiple dependent sub-tasks I_i^j equals to that of task S_i ; $C2$ denotes that the latency T_i of the computation-intensive task S_i that completes vehicle v_i cannot exceed the maximum tolerance latency $T_i^{\max}$ of task S_i ; $C3$ means that the mec_n server allocated to the vehicle v_i computing resources F_n^i, can not exceed the mec_n server ' s total computing resources F_n, and the mec_n server allocated to all offloaded vehicle computing resources and can not exceed the mec_n server ' s total computing resources F_n ; $C4$ indicates that the BS bandwidth allocation factor λ_i of vehicle v_i should be greater than 0 and less than 1, and the sum of BS allocation bands of all vehicle v_i is less than the total BS bandwidth B_1. $C5$ represents the sum of the data amount of sub-tasks and computation results offloaded by all vehicles to the mec_n server, which cannot exceed the total storage resource W_n of the mec_n server ; $C6$ represents the range of discrete variables.

V. Algorithm Design for the Problem Solution

It can be seen that the objective function has discrete integer variables such as U_i^j and Z_i^j, and continuous variables such as F_n^i and λ_i. Therefore, the optimization problem is a MINLP problem. At present, many studies use intelligent optimization algorithms to solve MINLP problems. PSO algorithm has memory and good convergence, but the particle is a single direction of information transmission, easy to fall into local optimum. Two-way transmission of information between chromosomes in GA algorithm has good global optimization effect, but slow convergence. Aiming at the objective function of this paper, the PSO and GA algorithm are improved, and an adaptive particle swarm genetic hybrid algorithm is proposed, which not only makes the algorithm have fast convergence ability, but also has better global search ability. In order to ensure that all the precursor tasks execute before it when the task is executed, this paper uses the DAG topology sorting algorithm to solve it. Different topological sorting will also lead to different results, so it is necessary to obtain all topological sorting results. The climbing search algorithm is used to local search for different sorting results to obtain the optimal results. The specific algorithm flow chart is described in Fig.5.

A. Initialization of Particle and Genetic Population

According to the objective function, it is necessary to solve the K, U, Z, F, W, B six matrices. Therefore, this paper improves the coding method of PSO algorithm and GA algorithm, and uses a multi-matrix coding method. The example of multi-matrix coding is shown in Fig.6. The particle population and the genetic population are composed of six large matrices KZ, UZ, ZZ, FZ, WZ, BZ, where N is the population size and k is the number of vehicles. To obtain the coding of the i particle/chromosome, only the corresponding lines $(i-1)*k+1$ to $i*k$ of each matrix need to be taken out and operated correspondingly.

Topological matrix KZ initialization is randomly selected from all topological of vehicle tasks. The random number of offloading strategy matrix U and MEC index matrix Z is initialized with in (0,1], and then mapped to integers, when the number of MEC server $n = 3$, the mapping relationship is shown as Eq.48 and Eq.49 , respectively.

$$U_i^j = \begin{cases} 0, & 0 < \mathrm{U}_i^j \le 0.25 \\ 1, & 0.26 < \mathrm{U}_i^j \le 0.5 \\ 2, & 0.51 < \mathrm{U}_i^j \le 0.75 \\ 3, & 0.76 < \mathrm{U}_i^j \le 1 \end{cases} \tag{44}$$

$$Z_i^j = \begin{cases} 1, & 0 < \mathrm{Z}_i^j \le (1/3) \\ 2, & (1/3) < \mathrm{Z}_i^j \le (2/3) \\ 3, & (2/3) < \mathrm{Z}_i^j \le 1 \end{cases}. \tag{45}$$

Algorithm 1 Select crossover and mutation by PMX and elitism scheme

INPUT: Genetic populations
OUTPUT: New populations after completion of selection, crossover and mutation
Initialization: Number of crossover points per time $NoPoint$=round$(k*sub-taskNum*Pc)$
$temp$=randperm$(k*sub-taskNum)$ Generate non-repeating random integers from 1 to $k*sub-taskNum$
Location of gene crossover $PoPoint$=temp$(:,1:NoPoint)$
$Emper=P_g$ global optimal individual
1 : **For** $j=1:(N/2)$ **do**
2 : $ParentU=UZ((2*j-1)*sub-taskNum+1:2*j*sub-taskNum,:)$
3 : $ParentZ=ZZ((2*j-1)*sub-taskNum+1:2*j*sub-taskNum,:)$
4 : Transform $ParentU$ and $ParentZ$ to 1 row with length $k*sub-taskNum$
5 : PMX crossover operation based on $NoPoint$ and $PoPoint$ to obtain new $ChildU$, $ChildZ$
6 : /* Mutation operations on $ChildU$, $ChildZ$
7 : **For** $m=1:k*sub-taskNum$ **do**
8 : $r=rand(1,1)$
9 : **If** $r\le P_m$
10 : $ChildU$, $ChildZ$=rand(1,1)$*(Xmax-Xmin)+Xmin$;
12 : $ChildU$, $ChildZ$ map back to integers
13 : **end if**
14 : After transforming $ChildU$, $ChildZ$ back into matrices and obtaining the corresponding F, W, B matrices
15 : **end For**
16 : **end For**
17 : Search the topological paths of the topological matrix with climbing search strategy
18 : Calculate the fitness
19 : **Update** New Matrix K
20 : Merging parent and sub-populations after crossover and mutation
21 : Sort the merged populations in ascending order according to fitness values
22 : Select the top N individuals as the new populations

The whole evolution process is mainly for the computation of offloading strategy matrix U and MEC numbering matrix Z, because after the offloading strategy is determined, the offloading MEC server can be selected according to the offloading results, and the resource allocation is carried out, so as the MEC server load balance.

B. Constructing the Fitness Function

The objective optimization model is to minimize the system cost and load balance with constraints. Using particle swarm genetic hybrid algorithm, the constraint problem needs to be transformed into a non-constrained problem, and the penalty function is only one of the commonly used methods in the optimization algorithm. This paper uses adaptive penalty function to constraint the objective function.

In the objective function, $C1$ and $C6$ constraints are processed in coding, and $C3$, $C4$ and $C5$ are bound to meet when allocating resources in bidding. Therefore, only punishment is needed for $C2$ constraints, so the fitness function can be obtained as follows:

$$fit=\mu\cdot H+(1-\mu)\cdot w\cdot\Psi+c(\rho)\cdot(\sum_{i=1}^{k}\max(0,T_i-T_i^{\max})) \tag{46}$$

$$c(\rho)=e^{\alpha(1-\rho)}-1 \tag{47}$$

$$\rho=\frac{f_{ok}}{f_{sum}} \tag{48}$$

Where the Ψ value is set as 1000, since the load balance is low, it is necessary to improve its proportion in the fitness function. $c(\rho)$ is the penalty coefficient, and ρ is the proportion of feasible solutions. ρ is inversely proportional to $c(\rho)$, and the larger ρ is, the more feasible solution is, and $c(\rho)$ should be reduced. When $c(\rho)$=0, it means that there is no feasible solution of the current population, and when $c(\rho)$=1, it means that the current population is all feasible. Because in the initial iteration, the number of feasible solutions in the population is low, and the penalty coefficient should be higher to guide the search into the feasible region.

C. Operations of particle swarm optimization and genetic algorithm

1) Particle velocity and position update: Large inertia weight makes particles have better global search ability, and small inertia weight is conducive to local search. Fixed inertia weight will fall into local optimization. Therefore, adaptive inertia weight is adopted in this paper, which can make particles move to the optimal position.

$$w^t=\begin{cases} w_{\min}+(w_{\max}-w_{\min})\cdot\frac{f_i^t-f_{\min}^t}{f_{avg}^t-f_{\min}^t}, f_i^t\le f_{avg}^t \\ w_{\max}, f_i^t>f_{avg}^t \end{cases} \tag{49}$$

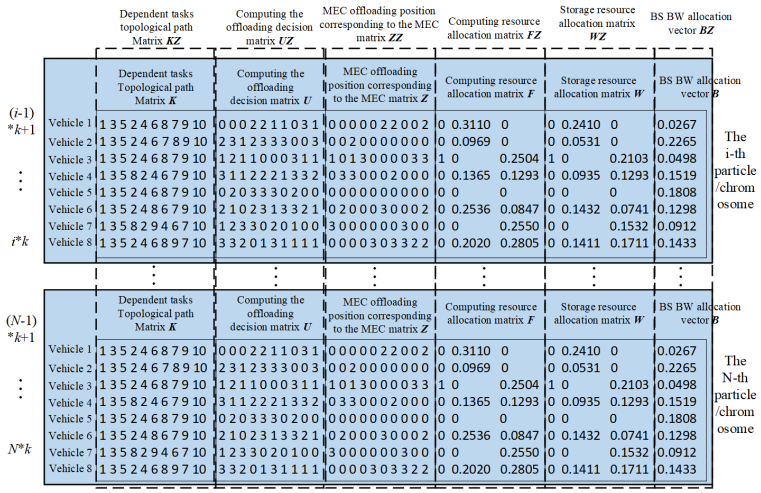

Fig. 6: Multi-matrix coding for problem solving

Where $w_{\min}$ and $w_{\max}$ are the initial minimum and maximum inertia coefficients, f_i^x is the fitness value of the first particle in the t iteration, f_{avg}^x is the average fitness value of all particles in the t iteration, and $f_{\min}^x$ is the minimum fitness value of all particles in the t iteration. The inertia weight decreases with the number of iterations, so that it has better global search ability in the early stage and better local search ability in the later stage.

Particle velocity and position update strategy is according to the following equations :

$$\begin{aligned} \mathrm{v}_i(t) =& \mathrm{w}^t \cdot v_i(t) + c_1 \cdot rand \cdot [p_i(t) - x_i(t)] \\ &+ c_2 \cdot rand \cdot [p_g(t) - x_i(t)] \end{aligned} \tag{50}$$

$$\mathrm{x}_i(t+1) = x_i(t) + \mathrm{v}_i(t) \tag{51}$$

Where $C1$ and $C2$ are learning factors, $p_i(t)$ and $p_g(t)$ are individual history optimal position and population global optimal position respectively.

2) Select crossover and mutation operations: In the GA algorithm, the crossover probability is too low, which affects the richness of the population in the iteration process, leading to slow convergence, but too large crossover probability will affect the inheritance of good individuals. Small mutation probability is not easy to produce new individuals, affecting population diversity, mutation probability will fall into local optimum. Therefore, this paper adopts the adaptive crossover probability and mutation probability. When the fitness value of the population is dispersed, it indicates that the population is rich, so the crossover probability and mutation probability should be reduced. On the contrary, when the fitness value of the population is relatively concentrated, it indicates that the richness of the population is not enough, so the crossover probability and mutation probability should increase.

Adaptive cross probability P_c:

$$P_c = \begin{cases} k_1 \cdot \frac{\arcsin\left(\frac{f_{\text{avg}}}{f_{\max}}\right)}{\pi/2}, & \arcsin\left(\frac{f_{\text{avg}}}{f_{\max}}\right) < \pi/6 \\ k_1 \cdot (1 - \frac{\arcsin\left(\frac{f_{\text{avg}}}{f_{\max}}\right)}{\pi/2}), & \arcsin\left(\frac{f_{avg}}{f_{\max}}\right) \geq \pi/6 \end{cases} \tag{52}$$

Adaptive mutation probability P_m:

$$P_m = \begin{cases} k_2 \cdot (1 - \frac{\arcsin\left(\frac{f_{\text{avg}}}{f_{\max}}\right)}{\pi/2}), & \arcsin\left(\frac{f_{\text{avg}}}{f_{\max}}\right) < \pi/6 \\ k_2 \cdot \frac{\arcsin\left(\frac{f_{\text{avg}}}{f_{\max}}\right)}{\pi/2}, & \arcsin\left(\frac{f_{avg}}{f_{\max}}\right) \geq \pi/6 \end{cases} \tag{53}$$

Where f_{avg} is the average fitness of the current population, $f_{\max}$ is the maximum fitness of the current population, k_1 and k_2 are constants in the range of (0,1]. Partial-Mapped Crossover (PMX) and monarch scheme are combined to select and cross operation. Firstly, the population is arranged in ascending order according to the fitness value, and the optimal individual is crossed with other even-numbered individuals. Two new individuals are generated each time. Secondly, after the crossover, the newly generated individuals are mutated to generate sub-groups, and then their fitness values are calculated. After that, they are merged with the parent group, and are arranged in ascending order according to the fitness values. The former N individuals are taken as new groups for the next operation.

VI. Simulation and Results Analysis

A. Simulation parameter setting

In this section, in order to verify the effectiveness of the algorithm in this paper. In the initial simulation, we consider a cellular network deployed within a circular area with a radius r = 1 Km, where the vehicles are randomly distributed in this cellular network area. MEC server are deployed within a RSU, which provide storage and computation resource. For simplicity, we only consider a BS with n=3 MEC servers, i.e., 3 RSUs, the number of vehicles k is initially set to 12, and the corresponding bandwidth and computing power settings are shown in Table 1. For the computational task, we consider the dot product operation, which is the most commonly used in deep learning, and randomly loop the dot product according to the amount of operations to generate the corresponding computational task. For comparison with other algorithm. Random algorithm, PSAO algorithm proposed in work [33] and JODTS

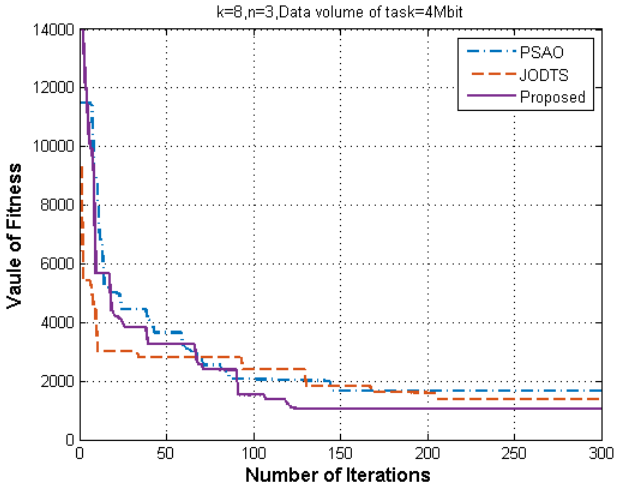

Fig. 7: Convergence performance of different algorithms

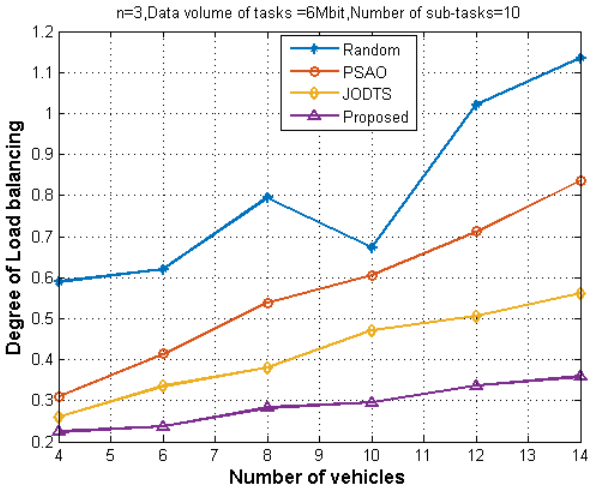

Fig. 9: The influence of the number of vehicles on the load balance

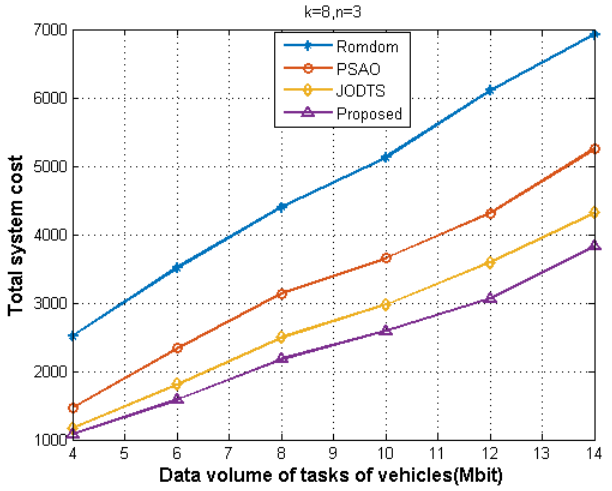

Fig. 8: The impact of tasks data size on the total cost of the system with different algorithms

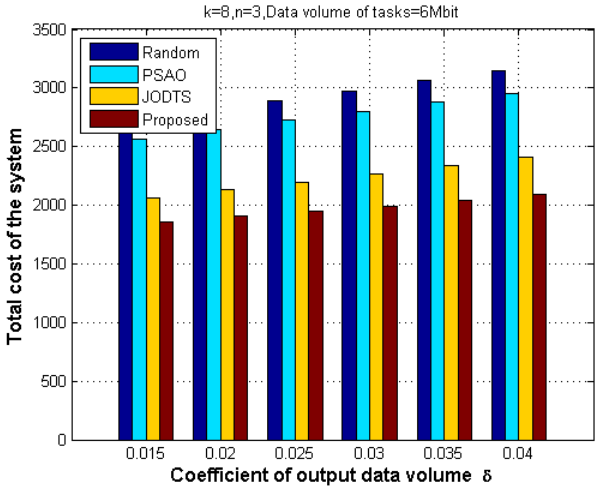

Fig. 10: The output data volume factor influence on the total system cost

algorithm proposed in work [11] are simulated and compared, The detail of compared algorithms is described as following:

Random algorithm : the dependent tasks are randomly offloaded to the local, MEC server, idle vehicle and cloud server for computation.

PSAO algorithm : Based on the offloading strategy of improved particle swarm algorithm, the computation task is offloaded to the local and the MEC server is calculated.

JODTS algorithm : Based on the offloading strategy of improved genetic algorithm, the computation task is offloaded to the local, MEC server and cloud server for computation.

B. Analysis of simulation results

Fig.7 shows the convergence performance of different algorithms. It can be seen that the convergence speed of this algorithm is the fastest, and it converges about 120 iterations. The convergence degree of PSAO is in the middle, and it converges about 140 iterations. The slowest convergence is JODTS algorithm, which converges about 210 iterations. In terms of global optimization, the algorithm in this paper is good, followed by JODTS algorithm, and PSAO algorithm is the worst. Because particle swarm algorithm is one-way to transmit information, genetic algorithm is two-way, so particle swarm algorithm convergence faster than genetic algorithm, but it is easy to fall into local optimum, so the particle swarm optimization ability is not strong genetic algorithm. This algorithm combines the characteristics of the two algorithms, and adds hill-climbing search strategy for local quadratic search, so the global optimization ability and convergence performance of this paper are improved.

Fig.8 shows that when the number of vehicles under DS and the number of MEC servers are fixed, the impact of computing tasks on the total cost of the system. It can be seen from the figure that with the increase of computation tasks, the total system cost of the four algorithms also increase. The total system cost of the proposed algorithm is the lowest, about 49.15% of Random algorithm, 68.10% of PSAO algorithm

TABLE I: Parameters of simulation

Parameters	Value
Number of vehicles to be offloaded under BS k	$4 \sim 14$
Number of MEC servers under BS n	3
Channel gain h_1 , h_2	0.8×10^{-4}, 1.2×10^{-4}
Channel gain of vehicle with MEC/ idle vehicle h_2	1.3×10^{-4}
Bandwidth between the car and BS B_1	8.4×10^{4} Hz
Bandwidth between vehicle and MEC/ idle vehicle B_2	4.8×10^{4} Hz
Transmitting power P_i^u, P_m^u, P_a^u, P_b^u, P_c^u	1.89,3.5,1.89,20,40 w
Computation power F_i,F_n,F_a,F_c	3×10^{5}, 8.9×10^{6},3×10^{5},5.6×10^{6}cycles/s
Noise power N^1, N^2	6×10^{-14},5×10^{-14} w
Task computation Complexity G_v , G_m,G_a,G_c	20∼40 cycles/bit
Maximum allowable latency of the task T_i^{max}	100∼300ms
Transmitting power P_i, P_m,P_a,P_c	5.31, 15,5.31,35 w
Coefficient of latency weight β, μ, α_1, α_2	0.8, 0.5, 0.7, 0.3
Coefficient of output data δ	0.015∼ 0.04
Relay latency between vehicles and idle vehicles t_{aw}	6∼8 ms
Transmission rate of Optical link R_f^c,R_f^b	20, 10 Mbps
Particle swarm size N	600
Learning factor c_1, c_2	2, 1.8
Maximum/ Minimum Velocity V_{min}, V_{max}	-0.18,0.18
k_1, k_2	0.53,0.42
Maximum/ Minimum X_{min},X_{max}	0, 1
Maximum iteration	300

and 87.62% of JODTS algorithm. And when the amount of computing tasks is more than 12, the increase is significantly increased, because when the amount of computing tasks is too large, resulting in data transmission and backhaul overhead, and MEC computing overhead increased significantly. It can be seen from the figure that the average latency of each vehicle in this algorithm is less than the maximum latency of the computation task, indicating that this algorithm can minimize the total system cost while meeting the maximum tolerance latency. Fig.9 shows the influence of the number of vehicles on the load balance when the number of MEC servers, the number of computation tasks and the number of sub-tasks are fixed under DS. It can be seen from the figure that the proposed algorithm has the smallest load balance and the smallest increase compared with the other three algorithms, indicating that the proposed algorithm can effectively balance the load of MEC server under DS. Due to the uneven distribution of vehicle mobility, the load balance of Random algorithm is the largest and fluctuates.

It can be seen from Fig.10 that when the number of vehicles, the number of MEC servers and the amount of computing tasks under DS are fixed, the total system cost increases as the output data volume factor δ increases. Compared with the other three algorithms, the proposed algorithm has the smallest total system cost. It can be seen that the output data volume has little effect on the total cost of the system, because the output data volume is relatively small, and the optical fiber is used between MEC servers, between MEC and BS, and between BS and cloud servers in this paper, so the bandwidth is large, so the output data volume has little effect on the total cost of the system.

Fig.11 shows the impact of the number of sub-tasks of vehicle tasks on the total cost of the system when the number of vehicles, the number of MEC servers and the amount of computation tasks are fixed under DS. It can be seen from the diagram that the total system cost of the four algorithms increase with the increase of the number of vehicle sub-tasks,

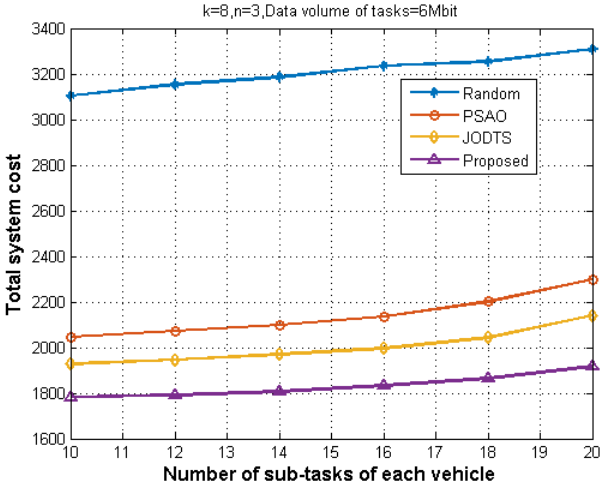

Fig. 11: The impact of the number of sub-tasks of vehicle tasks on the total cost of the system

and the increase is obvious when the number of sub-tasks exceeds 19. The reason is that when the number of sub-tasks increases, the dependencies between tasks become complex, which increases a lot of transmission overhead. And when the number of sub-tasks exceeds a certain number, the follow-up sub-task may wait for the completion of the previous sub-task computation, resulting in waiting latency. Compared with other algorithms, the algorithm in this paper has the smallest total system cost and the smallest increase, which can effectively deal with the dependence between sub-tasks.

The impact of two different coefficients on the value of the objective function is given in Fig.12. From the figure, we can find the effect of β for the system delay and the total system cost weight μ on the value of the objective function, and $1-\beta$ is defined as the energy consumption coefficient. It can be set according to the demand of computation intensive tasks and the state of vehicles. $1-\mu$ is defined as the weight of load

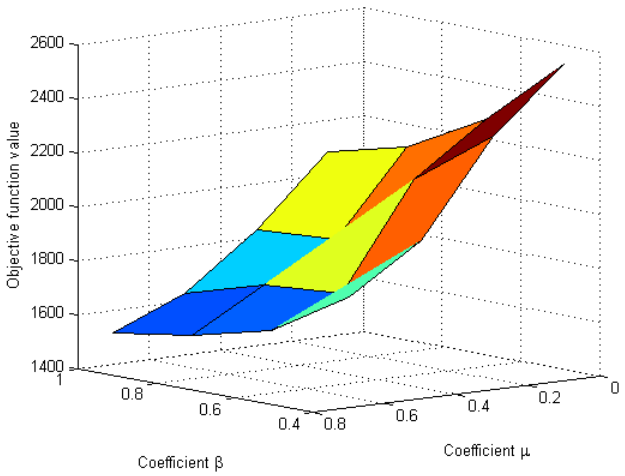

Fig. 12: The impact of the coefficients β and μ on the objective function values

balance in the objective function.

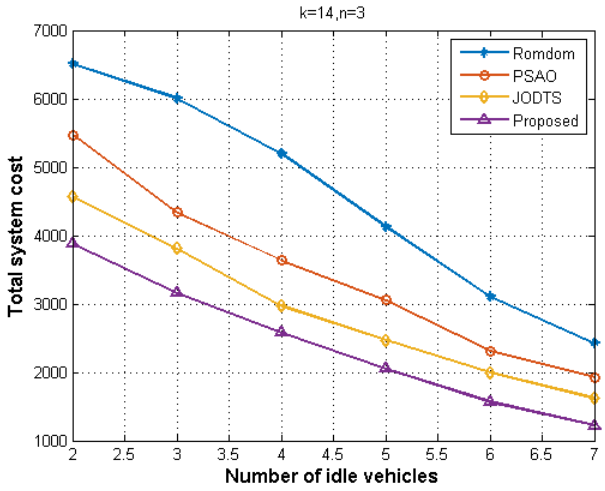

Fig. 13: The impact of the number of idle vehicles on the total cost of the system

Finally, we evaluated the impact of idle vehicles on the total system cost for different algorithms, as shown in Fig.13. From the figure, we can see that the total system cost with different algorithms decreases as the number of idle vehicles in the system increases, and our proposed algorithm maintains a low system cost for different idle vehicles in the system, which is due to the fact that our algorithm adopts an optimal computation task offloading strategy considering the timing and data dependencies of computational tasks.

VII. Conclusion

In this paper, in order to solve the problem of dependent latency caused by the dual dependence of timing and data between sub-tasks, the uneven distribution of vehicle mobility causes the load imbalance of MEC server, and in order to reduce latency and energy consumption. This paper proposes a joint computation offloading and resource allocation algorithm based on dual-dependent tasks. Considering that multiple vehicle dependent tasks are offloaded to multiple computing platforms at the same time, an adaptive hybrid particle swarm with genetic algorithm is proposed to jointly optimize offloading strategy, resource allocation and load balance. Through the simulation latency, it is verified that the proposed algorithm can reduce the total system cost while meeting the maximum latency, and effectively improve the load balance of the edge server cluster.

References

[1] W. Cao, Y. Wu, C. Chakraborty, D. Li, L. Zhao, and S. K. Ghosh, "Sustainable and transferable traffic sign recognition for intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, 2022. [Online]. Available: https://10.1109/TITS.2022.3215572

[2] Z. Guo, K. Yu, A. K. Bashir, D. Zhang, Y. D. Al-Otaibi, and M. Guizani, "Deep information fusion-driven poi scheduling for mobile social networks," *IEEE Network*, vol. 36, no. 4, pp. 210–216, 2022.

[3] Y. Li, H. Ma, L. Wang, S. Mao, and G. Wang, "Optimized content caching and user association for edge computing in densely deployed heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2130–2142, 2022.

[4] Z. Zhou, X. Dong, Z. Li, K. Yu, C. Ding, and Y. Yang, "Spatio-temporal feature encoding for traffic accident detection in vanet environment," *IEEE Transactions on Intelligent Transportation Systems*, 2022. [Online]. Available: https://doi.org/10.1109/TITS.2022.3147826

[5] Q. Zhang, K. Yu, Z. Guo, S. Garg, J. J. P. C. Rodrigues, M. M. Hassan, and M. Guizani, "Graph neural network-driven traffic forecasting for the connected internet of vehicles," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3015–3027, 2022.

[6] Z. Guo, K. Yu, K. Konstantin, S. Mumtaz, W. Wei, P. Shi, and J. J. P. C. Rodrigues, "Deep collaborative intelligence-driven traffic forecasting in green internet of vehicles," *IEEE Transactions on Green Communications and Networking*, 2022.

[7] S. Xia, Z. Yao, G. Wu, and Y. Li, "Distributed offloading for cooperative intelligent transportation under heterogeneous networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16 701–16 714, 2022.

[8] Z. Guo, K. Yu, Z. Lv, K.-K. R. Choo, P. Shi, and J. J. P. C. Rodrigues, "Deep federated learning enhanced secure poi microservices for cyber-physical systems," *IEEE Wireless Communications*, vol. 29, no. 2, pp. 22–29, 2022.

[9] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, "Efficient offloading for minimizing task computation delay of noma-based multiaccess edge computing," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3186–3203, 2022.

[10] Y. Lu, L. Yang, S. X. Yang, Q. Hua, A. K. Sangaiah, T. Guo, and K. Yu, "An intelligent deterministic scheduling method for ultra-low latency communication in edge enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2022.

[11] L. Zhao, H. Chai, Y. Han, K. Yu, and S. Mumtaz, "A collaborative v2x data correction method for road safety," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 951–962, 2022.

[12] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2019.2959410

[13] H. Wang, X. Li, H. Ji, and H. Zhang, "Federated offloading scheme to minimize latency in mec-enabled vehicular networks," in *IEEE Globecom Workshops, GC Wkshps 2018, Abu Dhabi, United Arab Emirates, December 9-13, 2018*. IEEE, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOCOMW.2018.8644315

[14] Y. Sun, K. Yu, A. K. Bashir, and X. Liao, "Bl-iea: A bit-level image encryption algorithm for cognitive services in intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2021.

[15] D. Peng, D. He, Y. Li, and Z. Wang, "Integrating terrestrial and satellite multibeam systems toward 6g: Techniques and challenges for interference mitigation," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 24–31, 2022.

[16] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 1, pp. 235–250, 2020. [Online]. Available: https://doi.org/10.1109/TWC.2019.2943563

[17] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5g iov architecture based on SDN and fog-cloud computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3832–3840, 2021. [Online]. Available: https://doi.org/10.1109/TITS.2020.3048844

[18] M. LiWang, S. Hosseinalipour, Z. Gao, Y. Tang, L. Huang, and H. Dai, "Allocation of computation-intensive graph jobs over vehicular clouds in iov," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 311–324, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2949602

[19] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, 2019. [Online]. Available: https://doi.org/10.1109/TVT.2018.2883156

[20] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12 175–12 186, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2020.3013990

[21] M. Zhu, Y. Hou, X. Tao, T. Sui, and L. Gao, "Joint optimal allocation of wireless resource and MEC computation capability in vehicular network," in *2020 IEEE Wireless Communications and Networking Conference Workshops, WCNC Workshops 2020, Seoul, Korea (South), April 6-9, 2020*. IEEE, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/WCNCW48565.2020.9124737

[22] T. Zhou, Y. Yue, D. Qin, X. Nie, X. Li, and C. Li, "Joint device association, resource allocation and computation offloading in ultra-dense multi-device and multi-task iot networks," *CoRR*, vol. abs/2112.05891, 2021. [Online]. Available: https://arxiv.org/abs/2112.05891

[23] Z. Ning, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Joint computation offloading, power allocation, and channel assignment for 5g-enabled traffic management systems," *IEEE Trans. Ind. Informatics*, vol. 15, no. 5, pp. 3058–3067, 2019. [Online]. Available: https://doi.org/10.1109/TII.2019.2892767

[24] B. Dab, N. Aitsaadi, and R. Langar, "Joint optimization of offloading and resource allocation scheme for mobile edge computing," in *2019 IEEE Wireless Communications and Networking Conference, WCNC 2019, Marrakesh, Morocco, April 15-18, 2019*. IEEE, 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1109/WCNC.2019.8885537

[25] Z. Chang, L. Liu, X. Guo, and Q. Sheng, "Dynamic resource allocation and computation offloading for iot fog computing system," *IEEE Trans. Ind. Informatics*, vol. 17, no. 5, pp. 3348–3357, 2021. [Online]. Available: https://doi.org/10.1109/TII.2020.2978946

[26] J. Ren, K. M. Mahfujul, F. Lyu, S. Yue, and Y. Zhang, "Joint channel allocation and resource management for stochastic computation offloading in MEC," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8900–8913, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2020.2997685

[27] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3226–3237, 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2021.3097754

[28] M. Gong and S. Ahn, "Computation offloading- based task scheduling in the vehicular communication environment for computation-intensive vehicular tasks," in *2020 International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2020, Fukuoka, Japan, February 19-21, 2020*. IEEE, 2020, pp. 534–537. [Online]. Available: https://doi.org/10.1109/ICAIIC48513.2020.9064975

[29] R. Chai, J. Lin, M. Chen, and Q. Chen, "Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D MEC systems," *IEEE Syst. J.*, vol. 13, no. 4, pp. 4110–4121, 2019. [Online]. Available: https://doi.org/10.1109/JSYST.2019.2921115

[30] H. Guo, J. Zhang, and J. Liu, "Fiwi-enhanced vehicular edge computing networks: Collaborative task offloading," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 45–53, 2019. [Online]. Available: https://doi.org/10.1109/MVT.2018.2879537

[31] Y. Zhang, M. Zhang, C. Fan, F. Li, and B. Li, "Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment," *EURASIP J. Adv. Signal Process.*, vol. 2021, no. 1, p. 33, 2021. [Online]. Available: https://doi.org/10.1186/s13634-021-00750-6

[32] W. Gong, L. Pang, J. Wang, and M. Xia, "Pso-based resource allocation in software-defined heterogeneous cellular networks," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 5, pp. 2243–2257, 2019. [Online]. Available: https://doi.org/10.3837/tiis.2019.05.001

[33] S. Scott-Hayward and E. Garcia-Palacios, "A PSO approach to resource allocation in wireless networks," in *35th International Conference on Telecommunications and Signal Processing, TSP 2012, Prague, Czech Republic, July 3-4, 2012*. IEEE, 2012, pp. 151–155. [Online]. Available: https://doi.org/10.1109/TSP.2012.6256271