Hierarchical Resource Management Framework and Multi-hop Task Scheduling Decision for Resource-Constrained VEC Networks

Xi Hu[1]*, Yicheng Zhao[1], Yang Huang[1], Chen Zhu[1], Jun Yao[1], and Nana Fang[1]
[1]Northeastern University at Qinhuangdao
Hebei, China
[e-mail: hux@neuq.edu.cn,
{2072098, 2072059, 2072099, 2172183, 2172154}@stu.neu.edu.cn]
*Corresponding author: Xi Hu

Received April 24, 2022; revised July 17, 2022; revised Septebmer 10, 2022; accepted October 22, 2022; published November 30, 2022

Abstract

In urban vehicular edge computing (VEC) environments, one edge server always serves many task requests in its coverage which results in the resource-constrained problem. To resolve the problem and improve system utilization, we first design a general hierarchical resource management framework based on typical VEC network structures. Following the framework, a specific interacting protocol is also designed for our decision algorithm. Secondly, a greedy bidding-based multi-hop task scheduling decision algorithm is proposed to realize effective task scheduling in resource-constrained VEC environments. In this algorithm, the goal of maximizing system utility is modeled as an optimization problem with the constraints of task deadlines and available computing resources. Then, an auction mechanism named greedy bidding is used to match task requests to edge servers in the case of multiple hops to maximize the system utility. Simulation results show that our proposal can maximize the number of tasks served in resource constrained VEC networks and improve the system utility.

Keywords: Vehicular Edge Computing (VEC), Multi-hop task scheduling, Resource management framework, Auction, Greedy matching.

This work was supported in part by the NSFC under Grant 61501102.

http://doi.org/10.3837/tiis.2022.11.010 **ISSN : 1976-7277**

1. Introduction

With the development of the Internet of Vehicles (IoV), more and more onboard applications are provided by smart vehicles, such as active driving assistance, road traffic monitoring, automatic management, and entertainment applications. Some of these applications require intensive computation and tight delay constraints, which are beyond vehicles' capabilities. Mobile edge computing (MEC) is an emerging but attractive computation framework. Different from cloud computing, MEC makes the task computing adjacent to the requestor, so it meets the aforementioned requirements of IoV applications [1]. Vehicular edge computing (VEC) is the combination of IoV and MEC. VEC can boost the development of an intelligent transportation system (ITS). It is a promising technology to meet the needs of these emerging applications by making rational use of the computing resources of a vehicle and network edges [2].

In VEC networks, a MEC server services all the vehicles covered typically [3]. This will lead to competition for the limited computing resource of MEC servers within vehicles [4]. As a result, task scheduling and resource allocation decisions are widely studied.

A Roadside Unit (RSU) equipped with an edge server can service vehicles within its coverage area, reducing the computational and backhaul pressure on the resource management center. Therefore, a three-layer resource scheduling architecture based on vehicles, RSUs, and resource management centers is constructed for the IoV edge network. But the existing MEC resource scheduling literature rarely pays attention to the system utility problem. Referring to [5], the edge cloud resources are allocated by auction, and the winner transaction price principle is adopted. However, this method does not consider the multi-product, multi-regional combination problem and the uncertainty of user bidding and ignores the two-way benefits of users and edge systems. Reference [6] proposes a novel caching architecture for edge system caching to achieve the purpose of energy-saving. But this type of method mainly considers the performance of the user and not the revenue of the margin service provider. Reference [7] considers the use of genetic decision-making to implement migration strategies for computing tasks with different priorities in the case of an unbalanced computing resource load, thereby improving the migration rate of edge computing. However, the full utilization of edge computing resources is not considered. Although resource scheduling has received a lot of attention in recent years, it usually ignores the collaboration between edge servers.

MEC resource allocation is one of the most important issues in MEC research. The authors of [8] developed a distributed resource sharing scheme for vehicle networks, where multiple vehicle-to-vehicle (V2V) links can share and reuse the spectrum allocated to vehicle-to-infrastructure (V2I) links to improve the system capacity and reliability. In [9], a joint scheme for unstable V2V links is proposed by considering a joint optimization model of transmission mode selection and resource allocation. In [10], a computational offloading and resource allocation scheme for time-varying multi-user MEC systems is proposed, and an algorithm based on centralized double deep Q-learning (DDQL) is proposed. A MEC based on a vehicle network was studied in [11], and a software-defined network (SDN) controller was proposed to collect global information on the network state. They utilize an SDN-based system to improve the efficiency of vehicle networks by optimizing offload node selection and resource allocation.

Considering that many mobile vehicles have some computational tasks that need to be processed, however, these vehicles lack sufficient computational resources. Therefore, vehicles need to rent computing resources from nearby RSUs, and to guarantee the quality of service,

the transmission delay of each task must be less than a deadline. In real life, there are multiple MEC servers on the real road. The resource management center needs to manage these servers reasonably and allocate MECs with insufficient resources to MECs with sufficient resources, which can not only improve network performance but also improve service quality [12].

For the resource allocation and pricing of edge computing systems, for edge providers, under limited resource conditions, users will bid for resources according to their budgets and costs to obtain better services, and resource providers will bid according to their bids [13-16]. Select users, deliver resources to users with higher bids, ensure user performance experience, and improve the overall revenue of edge systems. From a system point of view, edge service providers get more benefits from limited resources; Reference [17] studies the optimal service provider to maximize revenue under the constraints of quality of service (QoS) for all mobile users' resource allocation problems. Reference [18] proposed the concept of network service module market, using the method of stochastic process, to transform the problem of network resource allocation into the problem of resource buying and selling. The deterministic bidding method assumes that the user's bid is known, and the uncertainty of the user's bid leads to information lag in the resource auction and pricing process, which makes the deterministic auction method limited. From the user's point of view, users can Free choice of resources is required, and the quality of service can be guaranteed [19]. Therefore, the auction-based edge resource scheduling method adopted in this paper can effectively improve the overall system efficiency. There are many research results on system utility and resource management, which are well summarized in Reference [20].

There has been some research devoted to resource allocation and workload scheduling of edge cloud. Reference [21] applied the Markov approximation framework to solve the computational offload scheduling problem by using a distributed algorithm. Reference [22] proposes an iterative algorithm to solve joint communication and computational resource allocation problems. However, most of these studies do not address the problem of optimal matching between task scheduling and servers, and most studies on auction decision-making are concerned with maximizing benefits for service providers. Ignore the problem of maximizing system utility. For edge providers, under the condition of limited resources, users will bid for resources according to their budgets and costs to obtain better services. Resource providers select users based on their bids and deliver resources at higher prices. At the same time, it also ensures the user's performance experience and improves the overall revenue of the edge system. Reference [23] models computational offload scheduling as a mixed integer linear programming problem and designs quality of experience (QoE)-based node selection strategy based on the solution of the piecewise optimization problem. Referring to [24] to establish a global optimization problem, an uncertain offload scheduling algorithm is designed to minimize the expected delay. However, most of these studies study resource allocation from the perspective of workload scheduling, which is also different from considering resource competition from an economic perspective and ultimately achieves the goal of maximizing system utility. Referring to [25], a game-theoretic online algorithm is designed to solve the problem of offloading scheduling of computing tasks, and an online bin packing algorithm is used to calculate resource allocation. Although the literature [25] also designed a three-layer IoV model, uploading vehicle tasks to the cloud layer will cause a large delay, which is not suitable for the processing of delay-sensitive tasks. From a system perspective, edge service providers gain more benefits from limited resources; from a user perspective, users can freely choose the resources they need to ensure service quality. Since latency and server capacity constraints are important factors affecting user service quality, our goal is to optimize system

utility under the constraints of task request latency and available computing resources of the MEC server.

In VEC, several recent works focus on multi-hop offloading strategies based on vehicle coordination and server coordination. Reference [26] uses a vehicle-coordinated multi-hop offloading method to handle the computational task and proposes a semi-definite relaxation method with an adaptive adjustment process to solve the proposed optimization problem to obtain the corresponding unloading decision. Reference [27] proposed a cluster-based cellular-vehicle to everything (C-V2X) network collaborative task offloading scheme. This scheme discusses the cooperation between the MEC server and the vehicles with idle computing resources. However, the above literature does not consider the cooperation with the surrounding MECs and does not fully utilize the abundant computing resources of the MECs, which will lead to the waste of the surrounding MEC resources. Reference [28] proposed a collaborative method for offloading services to cars in a vehicle network based on MEC and cloud computing. The cloud-MEC collaborative computing offloading problem is established by collaboratively optimizing computing offloading decisions and computing resource allocation. Reference [29] proposes a Fiwi-enhanced in-vehicle edge computing network architecture to support the coexistence of remote cloud centers and lightweight edge servers connected to RSUs. Reference [30] studies the multi-hop computing offloading problem of the edge cloud computing model of the Industrial Internet of Things, and adopts the game theory method to realize the computing offloading of distributed perception quality of service (QoS). However, the above work focuses on the offloading of the cloud server and MEC server, but this will lead to an increase in latency, which is not conducive to the offloading of low latency tasks and the full utilization of MEC server resources. In order to solve the above problems, and fully schedule MEC server resources, this paper proposes a multi-hop MEC task scheduling decision, which is different from the traditional two-layer edge network hierarchical structure. Firstly, based on a two-level cluster VEC network, a three-layer resource management structure model is constructed to realize efficient resource management and computing offloading. Secondly, the computing offloading tasks are balanced to adjacent indirect connecting MEC servers through a multi-hop RSU-to-RSU (R2R) path. Thirdly, a nearby management node manages the multi-hop R2R computing offloading in the cluster.

This paper constructs a hierarchical model of the VEC network based on cluster class. In the model, multiple vehicles move within the coverage area of the RSU, and multiple RSUs cooperate to serve the vehicle. Under the constraints of service execution delay and limited storage of the MEC server, this paper adopts the GBMTS algorithm to maximize the system utility while satisfying the constraints of task request delay and available computing resources of the MEC server. First, we model the system utility maximization problem as a many-to-one weighted bipartite graph matching problem with multiple knapsack constraints and then propose the GBMTS decision algorithm to solve this problem. The main contributions of this work are as follows.

The main contributions of our work are as follows.

1) A hierarchical resource management system for VEC is proposed, which consists of a framework and an interacting protocol. In the framework, the hierarchical resource management structure and the main functional modules of each layer are designed. Then these modules cooperate for resource management with the interacting protocol. It should be noted that this system can be easily extended and applied to other MEC environments.

2) Considering the benefit of vehicle users and the cost of the MEC server, the task scheduling problem is modeled as a multi-knapsack restricted multi-to-one weighted bipartite graph matching under the delay constraints of tasks and the resource constraints of MEC servers.

Furthermore, to improve the utilization of resources and the success rate of computing requests, it's a multi-hop matching based on the cluster-based hierarchical structure.

3) To solve the afore matching problem, a greedy bid-winning multi-hop task scheduling decision algorithm is proposed based on the auction mechanism.

2. The hierarchical resource management framework and protocol for VEC Networks

VEC network is a typical resource-constrained environment, so effective resource management is essential to make full use of resources and improve service capabilities. As a result, a hierarchical resource management framework and an interacting protocol are proposed in this section.

2.1 The hierarchical resource management framework

Considering the practical environments of IoV and MEC, a typical cluster-based hierarchical structure of VEC networks is concluded as in **Fig. 1**. In this structure, the main part is the two-level cluster, based on which effective network management can be implemented. As shown in **Fig. 1**, the 1st-level cluster contains one head, i.e., gNB node in 5G or other base station node, and many members, i.e., vehicles and RSUs combined with MEC servers. The 2nd-level cluster also contains one head, i.e., the RSU node, and many members, i.e., vehicles. Furthermore, one 1st-level cluster contains multiple 2nd-level clusters, and the bidirectional wireless links exist between every neighbor cluster head pair, neighbor cluster head and member pair, and neighbor cluster member pair.

For realizing effective resource management in cluster-based VEC networks shown in **Fig. 1**, a hierarchical resource framework is designed. The detail of the framework is described in **Fig. 2**. The whole framework is divided into three layers, named layer0, layer1, and layer2, respectively.

1) Layer0: Consists of vehicles that are the offloading requesters and the task executors and contains three main modules, i.e., a local resource monitoring module for monitoring the resource usage of the vehicle itself, a task processing decision module for task division and offloading, etc. and a task computing module for executing tasks.

2) Layer1: Consists of MEC servers which are the executors of the offloading computing and the task migration, and contains four main modules, i.e., a local resource monitoring module for monitoring the resource usage of the MEC server itself, a resource allocation decision module for allocating the necessary resources for tasks or giving a task migration decision with insufficient resources, a task migration module for executing task migration and a task computing for executing tasks.

3) Layer2: Consists of base stations, such as gNBs, which are the resource management centers (RMCs) for making task scheduling decisions, and contains three main modules, i.e., a cluster resource monitoring module for monitoring the resource usages of all nodes in the cluster, a task scheduling strategy module for adopting different task scheduling strategies and a task scheduling decision module for making scheduling decisions according to strategies.

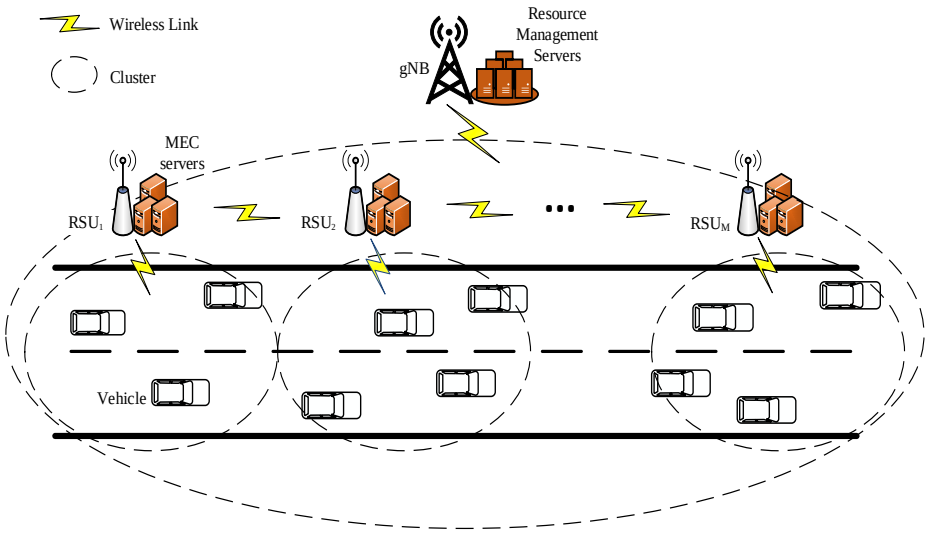


Fig. 1. The typical cluster-based hierarchical structure of VEC networks

2.2 The interacting protocol

The interacting protocol describes the interacting details among different modules of the framework. Therefore, the combination of the framework and the interacting protocol produces a workable hierarchical resource management system.

Next, we take the application of the system on VEC as an example to show the work process. However, it must be noted that this system can also be applied in other scenarios.

CASE 1: Pure Local Computing (PLC) on vehicles

STEP 1. A vehicle generates a task, and the local resource monitoring module of the vehicle tells the task processing decision module that the resource for computing the task is enough.

STEP 2. The task processing decision module decides to compute the task by the vehicle itself.

STEP 3. The task computing module of the vehicle computes the task locally.

CASE 2: Offloading Computing (OC) on MEC servers or vehicles

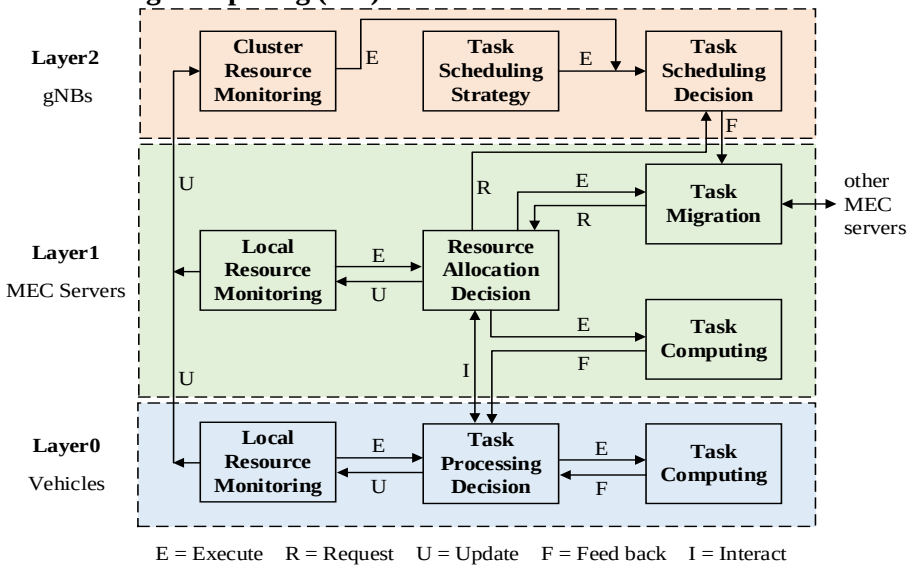


Fig. 2. The hierarchical resource management framework

In CASE 2, there are two subcases named Pure Offloading Computing (POC) and Hybrid Offloading Computing (HOC).

STEP 1. (on vehicle): A vehicle generates a task, and the local resource monitoring module of the vehicle tells the task processing decision module that the resource for computing the task is not enough, so the task processing decision module chooses one mode between POC and HOC.

STEP 2. (on vehicle): The task processing decision module interacts with the resource allocation decision module of the registered MEC server to execute offloading computing. If HOC is chosen, the task processing decision module is also responsible for segmenting the task into multiple subtasks and decides which are computed locally and which are offloaded to the MEC server.

STEP 3. (on MEC server): The resource allocation decision module of the registered MEC server receives the request, so it asks the local resource monitoring module whether there are enough resources for the task. If YES, it accepts the offloading computing request and goes to STEP 4 for computing. Otherwise, it requests the task scheduling decision module of gNB to schedule the task and goes to STEP 5.

STEP 4. (on MEC server): The task computing module of the registered MEC server computes the task offloaded by vehicles and then feeds the results back to the task processing decision of the vehicle.

STEP 5. (on gNB): The task scheduling decision module of gNB receives the task scheduling request, then it makes the scheduling decision according to the resource information from the cluster resource monitoring module and the scheduling strategy from the task scheduling strategy module.

STEP 6. (on gNB): The scheduling decision is fed back to the involved MEC server.

STEP 7. (on MEC server): When a MEC server receives feedback from gNB, it updates the migration rules in its task migration module. If it isn't the destination of migration, its task migration module will migrate the task received continuously. Otherwise, a task request will send to the resource allocation decision module, and then the task computing module will compute the task.

STEP 8. (on MEC server): After the task has been computed, the MEC server will feed the results back to the requesting vehicle.

STEP 9. (on vehicle): The requesting vehicle receives the result fed back. If POC is used, the result is the final one. If HOC is used, the task processing decision module integrates the results of different subtasks to get a meaningful and complete result.

3. Greedy Bid-based Multi-hop Task Scheduling Decision Algorithm

Based on the hierarchical resource management framework, a Greedy Bid based Multi-hop Task Scheduling decision algorithm named GBMTS is proposed in this section.

3.1 The delay model

There are three computing models used in MEC which are known as local computing, full offloading computing, and partial offloading computing, respectively. Because our work is about the multi-hop task scheduling decision, it is sufficient to only consider the offloading computing model. As a result, for simplicity, full offloading computing is taken as the example in our discussion without loss of generality.

In the full offloading computing model, the full task is offloading from the vehicle to the MEC server selected, which may be with a multi-hop mode in our discussions. Therefore, the whole delay experienced by the vehicle consists of the multi-hop transmission delay of the task, the computing delay on the computing MEC server, and the transmission delay of the result mainly.

Without loss of generality, we consider that there is only one task on each vehicle needed to be offloaded for the simplicity of discussion. Let $\mathrm{Task}_i = < D_i, A_i, T_i >$ denotes the task offloaded by the vehicle i (represented as Veh_i), where D is the amount of the task, A is the CPU cycles for computing the task, and T is the deadline required by the task. Furthermore, we suppose that the channels used by any neighbor nodes are bidirectional, and vehicles and MEC servers each have the same transmission power.

In the beginning, Task_i is transmitted Veh_i to its neighbor MEC server x (represented as Serv_x) in one hop, then Task_i is retransmitted by MEC servers in multi-hop mode until it reaches Serv_j. Therefore, the multi-hop transmission delay of the task $t_{i,j}^{\mathrm{FTran}}$ is computed as

$$t_{i,j}^{\mathrm{FTran}} = \frac{D_i}{r_{i,x}^{\mathrm{V2I}}} + \frac{D_i}{r_{x,y}^{\mathrm{I2I}}} \cdot (h-1) \tag{1}$$

where h is the hops from Veh_i to Serv_j, $r_{i,x}^{\mathrm{V2I}}$ and $r_{x,y}^{\mathrm{I2I}}$ are the wireless transmission rates used for V2I and I2I communications, respectively, which can be computed with Shannon's theorem as

$$r_{n,m} = B_{n,m} \log_2 (1 + \frac{p_n g_{n,m}}{\sigma^2}) \tag{2}$$

where $B_{n,m}$ is the channel bandwidth between two neighbor nodes <n, m>, p_n is the transmission power of the node n, $g_{n,m}$ is the channel gain and σ^2 is the noise power.

When Serv_j receive Task_i, it allocates some computing resources a_{ij} to compute the task. So, the computing delay t_{ij}^{Comp} is computed as

$$t_{ij}^{\mathrm{Comp}} = \frac{A_i}{a_{ij}} \tag{3}$$

The computation result is transmitted back Veh_i through a reverse path, so the transmission delay of the result $t_{j,i}^{\mathrm{RTran}}$ is computed as

$$t_{j,i}^{\mathrm{RTran}} = \frac{R_i}{r_{y,x}^{\mathrm{I2I}}} \cdot (h^{'} - 1) + \frac{R_i}{r_{x,i}^{\mathrm{I2V}}} \tag{4}$$

where R is the amount of the result.

Finally, the total delay t_{ij}^{Off} of Task_i on Serv_j is computed as

$$t_{ij}^{\mathrm{Off}} = t_{i,j}^{\mathrm{FTran}} + t_{ij}^{\mathrm{Comp}} + t_{j,i}^{\mathrm{RTran}} \tag{5}$$

According to [31], the amount of the computation result is much smaller compared to the task for many applications, e.g., face recognition. As a result, $t_{j,i}^{\mathrm{RTran}}$ can be ignored. Then, (5) can be simplified to

$$t_{ij}^{\mathrm{Off}} = t_{i,j}^{\mathrm{FTran}} + t_{ij}^{\mathrm{Comp}} \tag{6}$$

3.2 Problem formulation

In offloading computing, a MEC server with finite computation resources can't afford all of the computation tasks of vehicles. It results in the rejections of some task requests. Then these requests are forwarded by the MEC servers to the RMC (i.e., gNB) based on our proposed hierarchical resource management framework. When the RMC receives these requests, it will make a task scheduling decision that matches these tasks to the MEC servers in its cluster.

The task scheduling decision is a solution to the competition of the tasks for the limit and shareable computing resources, so it is modeled as a multiple-to-multiple auction which is defined in Definition 1.

Definition 1. The auction for the task scheduling decision is a multiple-to-multiple auction. In the auction, the RMC acts as the auctioneer. The tasks act as the bidders and bid for the computation resources of the MEC servers managed by the RMC in the 1st-cluster. Furthermore, due to the tasks can't be divided further, each task is matched to only one MEC server, but one MEC server can be matched by multiple tasks conversely.

It should be noted that different from the normal auction types, the auction in Definition 1 is a matching of multi-task to the multi-MEC server and have multiple different available auction results. Let B denote the bid set, which contains all the bids in the auction, b_{ij} ($b_{ij} \in \mathsf{B}$) denote the bid of Task_i for Serv_j and $S_{ij} = < b_{ij}, j >$ denote Task_i is successfully matched to Serv_j with b_{ij}, so any auction result is a matching set M which consists of all S_{ij} in the auction.

Definition 2. The utility of Serv_j is the profit earned by renting its computing resources to its matching tasks for offloading computing. Therefore, it can be computed as

$$U_j = \sum_{<b_{ij}, j|j>\in \mathsf{M}\ \mathsf{w}} (b_{ij} - A_i \times C_j) \tag{7}$$

where U_j and C_j denote the utility and the cost spent in one CPU cycle of Serv_j respectively.

Definition 3. The system utility is the total profit of all MEC servers involved in M. Therefore, it can be computed as

$$U_{\mathrm{system}} = \sum_{j\in \mathsf{M}} U_j \tag{8}$$

Where U_{system} denotes the system utility.

By substituting (7) into (8), the specific computation U_{system} is given by (9) as follows

$$U_{\mathrm{system}} = \sum_{<b_{ij}, j>\in \mathsf{M}\ \mathsf{w}} (b_{ij} - A_i \times C_j) \tag{9}$$

Definition 4. The optimal auction is the auction that produces the maximum system utility and the result of which is the optimal matching set.

Because the multi-hop task scheduling decision is equivalent to finding the optimal auction, the problem is converted to find the optimal matching set to maximize U_{system}. Then based on (9), the optimal problem of GBMTS is defined as follows.

Definition 5. Optimal Problem (OP) is finding the optimal matching set to maximize U_{system} under the constraints of delay and computation resources which is formulated as follows

$$\begin{aligned} OP(\mathsf{M}) &= \max\{\sum_{<b_{ij}, j>\in \mathsf{M}\ w} (b_{ij} - A_i \times C_j)\} \\ \text{s.t. } & \text{C1}: \mathsf{M} \subseteq \mathsf{B} \\ & \text{C2}: \sum_{<b_{ij}, j \mid j>\in \mathsf{M}} A_i \le W_j \\ & \text{C3}: t_{ij}^{\text{Off}} \le T_i \end{aligned} \tag{10}$$

Where W_j denotes the available computation resources of Serv_j. In (10), C1 shows that the matching set is produced from the bid set, C2 shows that the total computation resources needed by all tasks matched to Serv_j must be less or equal to the available computation resources Serv_j, C3 shows that the offloading computing delay of Task_i must be less or equal to its deadline.

3.3 GBMTS algorithm

In this section, the GBMTS algorithm is designed to resolve the optimal problem described in (10). The GBMTS algorithm can be divided into two consistent phases named pruning phase and greedy matching phase, respectively.

1) Pruning Phase

In the auction, each task generates a bid for every MEC server, but some of the bids are meaningless for the computation resource or the offloading computing delay of the MEC server is unavailable for the task. Therefore, in the pruning phase, these meaningless bids are removed from B by the RMC, and the details of pruning are shown in Algorithm 1.

Algorithm 1. Pruning

Input: B

Output: $\mathsf{B}_{\text{pruned}}$

1: **Initialization:** $\mathsf{B}_{\text{pruned}} = \phi$

2: **For** each $b_{ij} \in \mathsf{B}$ **Do**

3: **If** $t_{ij}^{\text{Off}} \le T_i$ **And** $A_i \le W_j$

4: $\mathsf{B}_{\text{pruned}} = \mathsf{B}_{\text{pruned}} \cup \{b_{ij}\}$

5: **End If**

6: **End For**

7: **Return** $\mathsf{B}_{\text{pruned}}$

As the result of pruning, the OP can be simplified by removing the constraint C3 from (10) as follows.

$$\begin{aligned} OP(\mathsf{M}) &= \max\{\sum_{<b_{ij}, j>\in \mathsf{M}\ w} (b_{ij} - A_i \times C_j)\} \\ \text{s.t. } & \text{C1}: \mathsf{M} \subseteq \mathsf{B} \\ & \text{C2}: \sum_{<b_{ij}, j \mid j>\in \mathsf{M}} A_i \le W_j \end{aligned} \tag{11}$$

2) Greedy Matching Phase

In the greedy matching phase, the RMC constructs a weighted bipartite graph $G = (V, E)$ based on the results of pruning, as shown in **Fig. 3**. The vertex set V consists of two disjoint

subsets: task subset V_{task} and MEC server subset V_{server} respectively. In the edge set E, for every $b_{ij} \in \mathrm{B}_{pruned}$, the corresponding $<\mathrm{Task}_i, \mathrm{Serv}_j>$ pair constructs an edge e_{ij} with the weight w_{ij} shown in (12).

$$w_{ij} = b_{ij} - A_i \times C_j \tag{12}$$

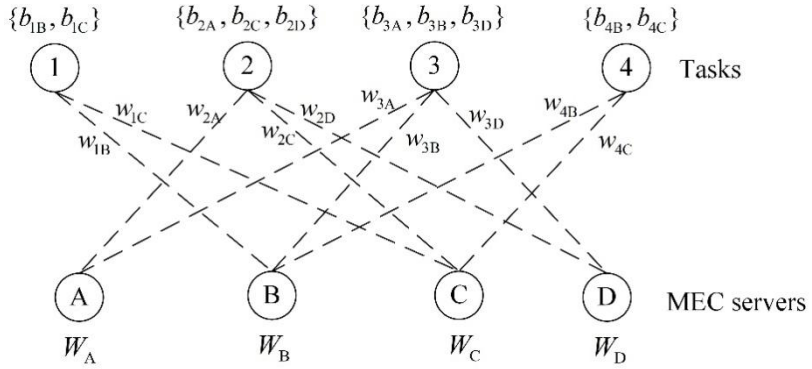


Fig. 3. An example of the weighted bipartite graph constructed by RMC

Greedy matching is an iterative greedy selection process. In each iteration, one edge with the maximum weight is selected. That is to say, one matching pair is found. When the greedy matching ends, all selected matching pairs construct a solution set which is the optimal matching set of OP. The details of greedy matching are shown in Algorithm 2.

Algorithm 2. Greedy Matching

Input: E

Output: Ω

Initialization: $\Omega = \phi$

While $E \neq \phi$ **Do**

  Select the edge with the maximum weight in E

  If multiple edges are selected **Then**

    Select an edge randomly

  Let e_{ij} be the selected edge

  If $A_i \leq W_j$ **Then**

    $\Omega = \Omega \cup \{b_{ij}\}$

    $W_j = W_j - A_i$

    Delete all edges related to Task_i from E

  Else

    $E = E - \{e_{ij}\}$

  End If

End While

Return Ω

3.4 The computation efficiency of GBMTS algorithm

GBMTS algorithm consists of two sequential sub-algorithms: Algorithm 1 and Algorithm 2. Considering the case that there are *n* tasks needed to be matched to *m* servers by the RMC. In Algorithm 1, it will iterate every bid in β for pruning, so the time complexity of Algorithm 1 is O(*nm*). In Algorithm 2, there are *nm* edges in the worst case, and it will iterate every edge to match tasks to servers, so the time complexity of Algorithm 2 is O(*nm*) too. Therefore, the time complexity of GBMTS algorithm equals to O(*nm*)+O(*nm*)= O(*nm*), that is GBMTS algorithm can be conducted in polynomial time.

4. Simulation and Result Analysis

4.1 Simulation Parameter Setting

The proposed GBMTS algorithm is simulated with *Matlab*, and the specific simulation parameters are shown in **Table 1**.

Table 1. simulation parameters

Parameters	**Value**
Vehicle computing capability a_i^l	[1,2] GHZ
Vehicle transmitting power p_i	[1,2] W
Computation input data size d_i	[10,100] KB
Noise power σ^2	-60dBm
Bandwidth B_j	[1,3] MHz
Number of CPU cycles required to complete the task	[20,80]
Capacity of the MEC server L_j	[60,200]
Transmitted power of the RSU p_j^c	[10,15] w
The channel gain h_{ij}	[1,3]
MEC server cost C_j	[0.1,1]
The number of vehicle tasks *L*	[5,30]
The number of MEC servers *M*	[5,15]

4.2 Results and Analysis

In the simulation, a benchmark algorithm named HBMTS and a centralized heuristic greedy offloading (CHGO) algorithm [29] are used to verify the effectiveness of our GBMTS algorithm.

- The highest bid based multi-hop task scheduling (HBMTS) decision algorithm works in a distributed mode, each vehicle generates bids for every MEC server in three hops. When a MEC server receives the bids from vehicles, it sorts all requests, the delay constraint of which can be satisfied, in a descending order based on the bids. Then the request with the highest bid will be replied. If a vehicle receives one reply, it offloads the task directly, else it chooses a MEC server randomly to offload. This process iterates until all requests are accepted or the delay constraint of task expires. In HBMTS, all interchanges are process in multi-hop R2R mode.

- The centralized heuristic greedy offloading (CHGO) algorithm adopts a centralized management mode. The management center matches tasks to MEC servers based on the minimum processing delay. In each iteration, a task is matched and offloaded to the MEC server which can achieve the minimum processing delay through multi-hop R2R path. This process iterates until all the vehicles have made their offloading decisions.

Fig. 4 shows the situation in which the utility value of the system changes with the increase of the number of requesting vehicles in the three cases of the number of MEC servers M=5, M=10, and M=15. With the increase in the number of requesting vehicles and the number of MEC servers, the available system utility increases. When M=5 and M=10, as a result of the limitation of MEC server resources, along with the increase of the number of vehicles is competition for resources, get service vehicles, the less instead, when the number of vehicles is greater than 25, satisfy the constraint conditions of the request has been assigned to complete basic vehicle, even if the number of vehicles increases, the winner of the number basically remains unchanged, decision convergence.

Fig. 5 shows that the system service rate increases with MEC servers when L=10, L=20, and L=30. As shown in the figure, with the increase in the number of MEC servers and vehicles, the system service rates of the three algorithms will also increase accordingly. Among them, the GBMTS algorithm increases the fastest because the GBMTS algorithm can improve the system service by increasing the number of MEC servers. Speed, so that more vehicles can be served.

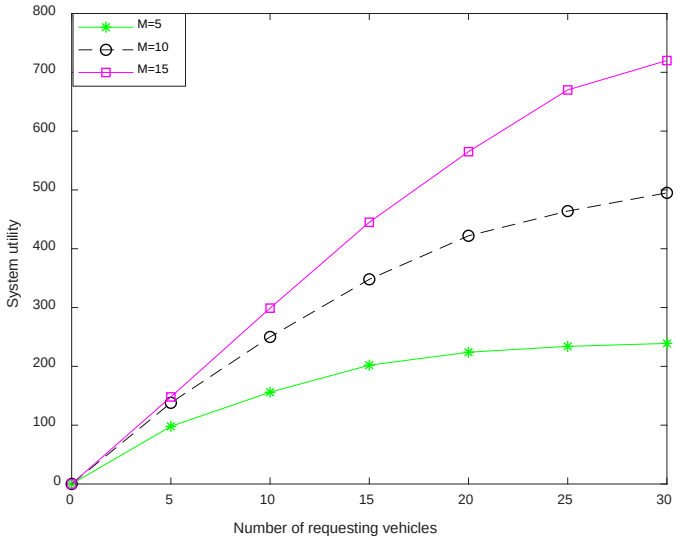


Fig. 4. System utility vs. Number of requesting vehicles with different MEC servers.

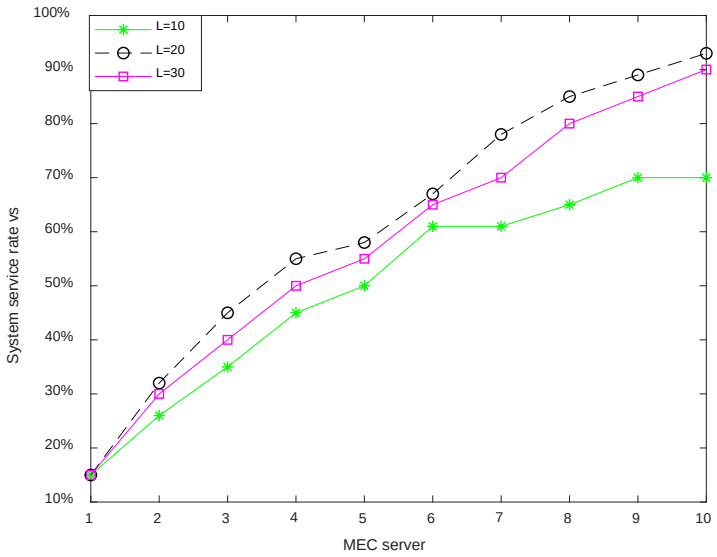


Fig. 5. System service rate vs. MEC servers with different vehicles.

Fig. 6 shows the changes in system utility caused by changes in the number of different MEC servers. As shown in the figure, as the number of edge servers increases, the available system utility values show an upward trend. Among them, the GBMTS decision is better than the lowest delay multi-hop task scheduling decision and the highest bid multi-hop task scheduling decision. Since the minimum delay decision only considers the optimal time delay and ignores the cost of the server, the system utility value is the lowest. The CHGO algorithm considers the delay and only needs to satisfy the CHGO decision. But it ignores the number of resources required by the vehicle itself. The HBMTS algorithm considers the optimal bidding, but it ignores the delay, so the system utility value is lower than that of GBMTS.

Fig. 7 shows the variation of vehicle winners due to the different number of MEC servers. As shown in the figure, with the increase of MEC servers, the number of winning vehicles also increases. Set the request vehicles L=20 and the number of MEC servers M=10. With the increase in the number of edge servers, the available system utility values show an upward trend. Among them, the GBMTS decision is better than the lowest delay multi-hop task scheduling decision and the highest bid multi-hop task scheduling decision. The CHGO algorithm only selects the optimal delay, ignoring the service to most vehicles. The HBMTS algorithm only considers the bidding of vehicles and chooses to ignore the vehicles that do not meet the bidding because the number of services is less than GBMTS.

Fig. 8 shows that when the number of requesting vehicles changes and the MEC server is fixed at M=10, the system utility values obtained by using different decisions are also different. With the increase in the number of requesting vehicles, the system utility values are increasing all the time. The decision proposed in this paper increases the fastest, and the system utility values obtained are also the most. When the number of requesting vehicles reaches 20, the system utility values obtained by using different multi-hop task scheduling decisions are the same, due to the limited capacity of resources, the system utility value increases slowly compared with before. However, after the number of vehicles reaches 20, the system utility value of the proposed decision also increases faster than the other two decisions. This is because the proposed decision can find the optimal node to serve the vehicles and realize the optimal allocation of resources under the condition of limited resources.

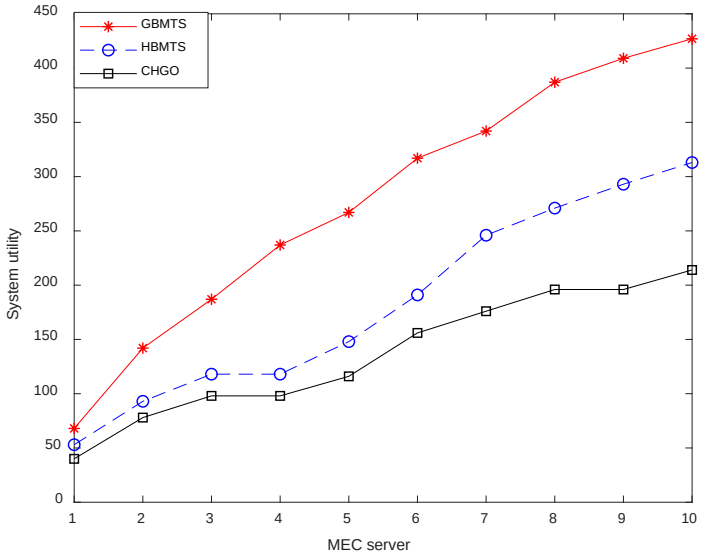


Fig. 6. System utility vs. MEC servers with different algorithms

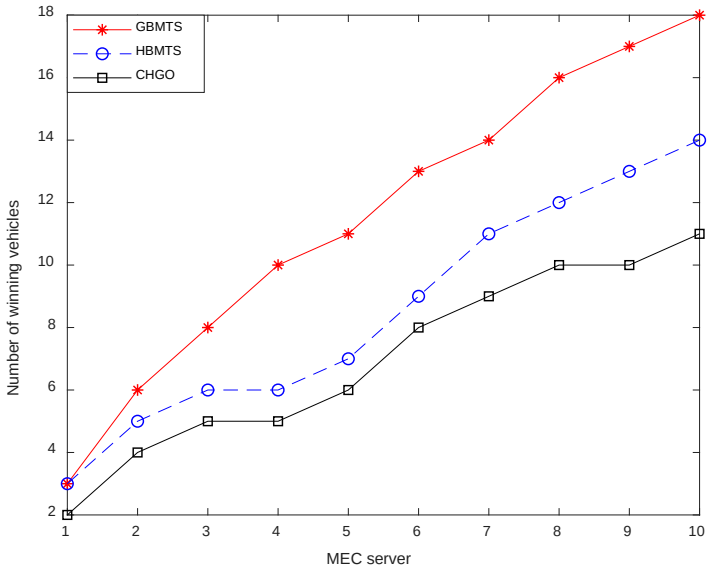


Fig. 7. Number of winning vehicles vs. MEC servers with different algorithms

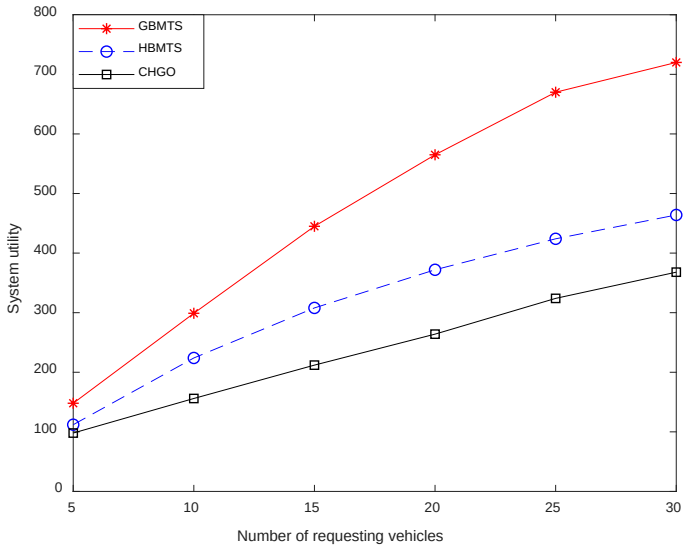


Fig. 8. System utility vs. Number of requesting vehicles with different algorithms.

Fig. 9 shows that when L=30, the average delay of all algorithms decreases as the number of MEC server increases, and HBMTS always has the largest delay for all scenes. Moreover, when the number of MEC servers is small (<4 in our simulations), CHGO has the smallest delay, but as it increases, GBMTS gets the smallest delay. This is because the matching mechanism of GBMTS can make use of the resources of MEC servers more effectively under the restriction of task deadlines.

Fig. 10 shows that when M=5, the average delay of all algorithms increases as the number of requesting vehicles increases, and HBMTS has the largest delay for all scenes. Moreover, when the number of requesting vehicles is more than 14, GBMTS gets the smallest delay for it can achieve more effective offloading decisions by considering the delay and available resources of MEC servers. This is further illustrated that GBMTS is more effective in an actual VEC network.

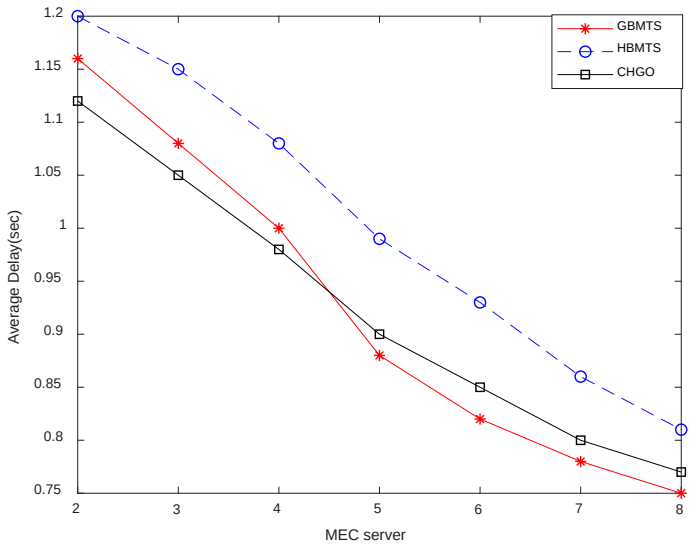


Fig. 9. Average Delay vs. MEC servers with different algorithms.

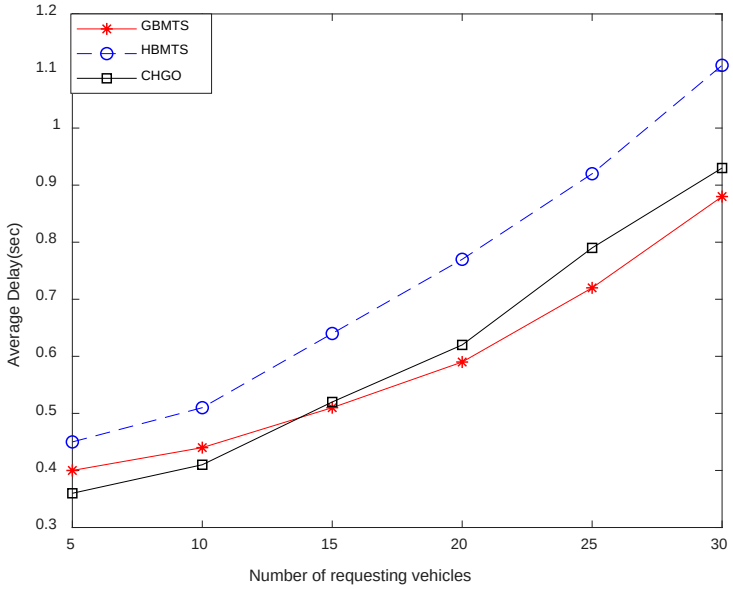


Fig. 10. Average Delay vs. Number of requesting vehicles with different algorithms.

5. Conclusion

In this paper, we consider the problem of scheduling tasks for time-sensitive vehicle users by auction mechanism and then allocating resources by multi-hop MEC matching the optimal MEC server so as to avoid overwork and waste of resources in any one MEC. This paper puts forward a kind of general applicable to large-scale car side end two-stage cluster at the edge of the network structure model. Within the cluster will task scheduling with MEC server model matching problem into multiple knapsack restrictions on a weighted binary map-matching problem, the GBMTS decision is proposed to solve the matching problem to achieve the rational allocation of resources, And the task scheduling path is planned to realize multi-hop offloading of the task. The simulation results show that the GBMTS decision can increase the number of serviced vehicles in the edge computing network of vehicle linkage and maximize the system utility under the condition of satisfying the time delay and capacity constraints.

References

[1] L. L. Wang, J. S. Gui, X. H. Deng, "Routing Algorithm Based on Vehicle Position Analysis for Internet of Vehicles," *IEEE Internet of Things Journal.*, vol. 7, no. 12, pp. 11701-11712, jun. 2020. Article (CrossRef Link)

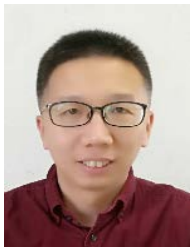
[2] J. Xu, L. Chen and S. Ren, "Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing," *IEEE Transactions on Cognitive Communications and Networking.*, vol. 3, no. 3, pp. 361-373, jul. 2017. Article (CrossRef Link)

[3] C. Lee and A. Fumagalli, "Internet of Things Security - Multilayered Method For End to End Data Communications Over Cellular Networks," in *Proc. of 2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019. Article (CrossRef Link)

[4] H. Li, X. Li, M. Zhang, and B. Ulziinyam, "Multicast-oriented task offloading for vehicle edge computing," *IEEE Access.*, vol. 8, pp. 187373-187383, Oct. 2020. Article (CrossRef Link)

[5] A. Kiani, N. Ansari, "Toward hierarchical mobile edge computing: an auction-based profit maximization approach," *IEEE Internet of Things Journal.*, vol. 4, no. 6, pp.2082-2091, Dec. 2017. Article (CrossRef Link)
[6] G. S. Li, Q. Y. Lin, J. H. Wu, "Dynamic computation offloading based on Graph partitioning in Mobile Edge Computing" *IEEE Access*, vol. 7, pp.185131-185139, Dec. 2019. Article (CrossRef Link)
[7] J. Cheng, D. Guan, "Research on task-offloading decision mechanism in mobile edge computing-based Internet of Vehicle," *EURASIP Journal on Wireless Communications*, pp. 2-14, Apr. 2021. Article (CrossRef Link)
[8] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019. Article (CrossRef Link)
[9] X. Zhang, M. Peng, S. Y an, and Y. Sun, "Deep-reinforcement-learning-based mode selection and resource allocation for cellular V2X communications," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6380–6391, Jul. 2020. Article (CrossRef Link)
[10] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022. Article (CrossRef Link)
[11] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Communications.*, vol. 17, no. 5, pp. 266-283, May. 2020. Article (CrossRef Link)
[12] Y. Liu, H. M. Yu, Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology.*, vol. 68, no. 11, pp. 11158-11168, Nov. 2019. Article (CrossRef Link)
[13] L. Tan, R. Hu, "Mobility-aware edge caching and computing in vehicle networks: a deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10190-10203, Nov. 2018. Article (CrossRef Link)
[14] G. H. Qiao, S. P. Leng, K. Zhang, "Collaborative Task Offloading in Vehicular Edge Multi-Access Networks," *IEEE Access*, vol. 8, pp. 51-59, Oct. 2020. Article (CrossRef Link)
[15] T. Feng, B. Wang, H. -t. Zhao, T. Zhang, J. Tang and Z. Wang, "Task distribution offloading algorithm based on DQN for sustainable vehicle edge network," in *Proc. of 2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021. Article (CrossRef Link)
[16] C. M. Xu, S. H. Liu, C. Zhang, "Multi-agent Reinforcement learning Based Distributed Transmission in Collaborative Cloud-Edge Systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1658-1672, Jan. 2021. Article (CrossRef Link)
[17] X. Guo, R. Singh, T. Zhao and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *Proc. of 2016 IEEE International Conference on Communications (ICC)*, 2016. Article (CrossRef Link)
[18] K. Zhang, Y. Mao, S. Leng, S. Maharjan and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. of 2017 IEEE International Conference on Communications (ICC)*, 2017. Article (CrossRef Link)
[19] D. T. Hoang, D. Niyato and P. Wang, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *Proc. of 2012 IEEE Wireless Communications and Networking Conference (WCNC)*, 2012. Article (CrossRef Link)
[20] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, Aug. 2017. Article (CrossRef Link)
[21] W. He, S. Guo, Y. Liang, and X. Qiu, "Markov approximation method for optimal service orchestration in IoT network," *IEEE Access*, vol. 7, pp. 49538–49548, Apr. 2019. Article (CrossRef Link)
[22] J. Zhang et al., "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018. Article (CrossRef Link)

[23]Z. Wang, S. Zheng, Q. Ge, K. Q. Li, "Online Offloading Scheduling and Resource Allocation Algorithms for Vehicular Edge Computing System," *IEEE Access*, vol. 8, pp. 52428-52442, Mar. 2020. Article (CrossRef Link)
[24]Y. Cao and Y. Chen, "QoE-based node selection strategy for edge computing enabled Internet-of-Vehicle(EC-lov)," in *Proc. of 2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1-4, Mar. 2018. Article (CrossRef Link)
[25]A. Ebra and M. Maier, "Distributed cooperative computation offloading in multi-access edge computing fiber-wireless networks" *Optics Communications*, vol. 452, pp. 130-139, Dec. 2019. Article (CrossRef Link)
[26]L. Liu, M. Zhao, M. A. Jan, A. Tah, "Mobility-Aware Multi-Hop Task Offloading for Autonomous Driving in Vehicular Edge Computing and Networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1-14, Jan. 2022. Article (CrossRef Link)
[27]M. Sal, P. Fan, G. Liu, "A cluster-based cooperative computation offloading scheme for C-V2X networks," *Ad Hoc Networks.*, vol. 132, pp. 1-12, Apr. 2022. Article (CrossRef Link)
[28]J. Zjao, Q. Li, Y. Gong, K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, pp.7944-7956, Jun. 2019. Article (CrossRef Link)
[29]H. Guo, J. Zhang, J. Liu, "Fiwi-enhanced vehicular edge computing networks: Collaborative task offloading," *IEEE Vehicular Technology Magazine.*, vol. 14, pp.45-53, Mar. 2019. Article (CrossRef Link)
[30]Z. Hong, W. Chen, H. Huang, S. Guo, Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, pp. 2759-2774, Jul. 2019. Article (CrossRef Link)
[31]K. Zhang, Y. Mao, S. Leng, Y. He and Y. ZHANG, "Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36-44, Apr. 2017. Article (CrossRef Link)



Xi Hu received the B.Eng. degree in Electronic Information Science and Technology from Liaoning University, Shenyang, China, in 2003, the M.S. and Ph.D. degrees in Communication and Information System from Northeastern University, Shenyang, China, in 2008 and 2011, respectively. He is currently an associate professor with Computer Center, Northeastern University at Qinhuangdao, Hebei, China. His current research interests include intelligent transportation systems, Internet of vehicles, edge/fog computing, and edge intelligence.

Yicheng Zhao received the B.Eng. degree in School of Physics and Optoelectronic Engineering from Weifang University, China, in 2020. He is currently pursuing the Master degree with the School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Hebei 066004, China. His current research interests include Internet of vehicles and resource scheduling based on resource cluster.

Yang Huang received the B.Eng. degree in School of Electronic Information Engineering from Anhui Polytechnic University, China, in 2019. He is pursuing the Master degree with the School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Hebei 066004, China. His current research interests include mobile edge computing and deep learning.

Chen Zhu received the B.Eng. degree in School of Electrical and Information Engineering, Anhui University of Technology, China, in 2019. He is currently pursuing the Master degree with the School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Hebei 066004, China. His current research interests include Internet of vehicles and load balancing.

Jun Yao received the B.Eng. degree in School of Electronic Information Engineering from Anhui Polytechnic University, China, in 2021. He is pursuing the Master degree with the School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Hebei 066004, China. His current research interests include mobile edge computing and deep learning.

NaNa Fang received the B.Eng. degree in School of Electronic Information Engineering from Anhui Polytechnic University, China, in 2021. He is pursuing the Master degree with the School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Hebei 066004, China. His current research interests include mobile edge computing.