NOMA-Based Task Offloading and Allocation in Vehicular Edge Computing Networks

Shuangliang Zhao[1,3], Lei Shi[1,3(✉)], Yi Shi[2], Fei Zhao[1,3], and Yuqi Fan[1,3]

[1] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China
shilei@hfut.edu.cn
[2] Department of ECE, Virginia Tech, Blacksburg, VA 24061, USA
[3] Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei 230009, China

Abstract. Vehicular Edge Computing (VEC) is envisioned as a promising approach to process explosive vehicle tasks. In the VEC system, vehicles can choose to upload tasks to nearby edge nodes for processing. This approach requires an efficient communication method, and Non-Orthogonal Multiple Access (NOMA) can improve channel spectrum efficiency and capacity. However, in the VEC system, the channel condition is complex due to the fast mobility of vehicles, and the arrival time of each task is stochastic. These characteristics greatly affect the latency of tasks. In this paper, we adopt a NOMA-based task offloading and allocation scheme to improve the VEC system. To cope with complex channel conditions, we use NOMA to upload tasks in batches. We first establish the mathematical model, and divide the offloading and allocation of tasks into two processes: transmission and computation. Then we determine appropriate edge nodes for transmission and computation according to the position and speed of vehicles. We define the optimization objective as maximizing the number of tasks completed, and find that it is an integer nonlinear problem. Since there are more integer variables, this optimization problem is difficult to solve directly. Through further analysis, we design Asymptotic Inference Greedy Strategy (AIGS) algorithm based on heuristics. Simulation results demonstrate that our algorithm has great advantages.

Keywords: VEC · NOMA · Task offloading · Task allocation · Collaborative processing

The work is supported by the major science and technology projects in Anhui Province, NO. 202003a05020009 and the innovation foundation of the city of Bengbu, Grant No. JZ2022YDZJ0019.

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 343–359, 2022.
https://doi.org/10.1007/978-3-031-24383-7_19

1 Introduction

With the rapid development of the Internet of Things (IoT), a large number of intelligent devices and other smart IoT devices are flooding into wireless networks [1]. Various smart applications have been flourishing, such as smart city, smart manufacturing, automatic driving [2,3], etc. Moreover, The Internet of Vehicles (IoV) has received a lot of attention in recent years, and the rapid development of IoV has also brought some challenges that need to be solved. Most vehicular applications are usually latency critical, and solving this problem requires strong processing capability. In addition, because of the extreme scarcity of wireless resources, large amounts of IoV device need to allocate wireless resources more efficiently [2,5]. Edge Computing (EC) has been regarded as a promising technology to reduce latency [4], and VEC is the application of EC in IoV [25]. Meanwhile, the development of NOMA can effectively utilize wireless resources and improve communication quality [6–8]. The combination of these two technologies has greatly contributed to solving the challenges in IoV [9–11].

In EC, Edge Nodes (ENs) can efficiently allocate communication and computing resources to users [12,14]. Compared to traditional cloud server, ENs are closer to users, so they can give feedback to requests in time [15]. Moreover, NOMA can effectively improve the channel utilization [13]. Therefore, the application of NOMA-assisted ENs has gradually become a mainstream scheme [16–20]. In addition, this scheme can achieve secure power allocation and reduce processing latency in industrial applications [21,22]. And using Unmanned Aerial Vehicle (UAV) to join this scheme can also achieve great results, but this approach also brings additional power consumption and scheduling problems [23,24].

In VEC, vehicles have a higher speed of movement and the requirements for communication are more demanding [26]. NOMA can increase the task of simultaneous transmission and improve the service quality of the system [27]. However, adopting NOMA technology in VEC also presents some challenges. First, since the speed of vehicle is fast, it is very critical to determine a suitable edge node in transmission process. Second, NOMA has a threshold requirement for Signal to Interference plus Noise Ratio (SINR) [14], and determining the order of task transmission has an important role in transmission process. Third, after uploading the task, the vehicle may leave the current edge node, so it is very important to select the appropriate edge node to process the task for the improvement of system performance, etc. These considerations motivate the study of this article and the main contributions of our work are summarized as follows,

1) For the high-speed mobility of the vehicle, we select the appropriate edge node to upload the task according to the location and speed of the vehicle. In this way, we can ensure that the task can be successfully uploaded to the edge node. After the task is uploaded, we determine the appropriate edge node to compute the task according to the location of the vehicle and the load of the edge nodes.
2) In order to increase the number of tasks transmitted simultaneously, we use NOMA to transmit tasks in the transmission process. Moreover, we transmit tasks in batches according to the position of the vehicle and the size of the task, In this way, the communication quality of NOMA can be improved.

3) For the optimization problem, we design the Asymptotic Inference Greedy Strategy (AIGS) algorithm to deal with situations in real roads. We execute many simulation experiments by simulating real-time road conditions, and the results show that our algorithm has great advantages.

The remainder of this article is organized as follows. Related works are presented in Sect. 2. Our system model and problem formulation are described in Sect. 3. Section 4 introduces our algorithm. The simulation and experiments are provided in Sect. 5. Finally, this paper is concluded in Sect. 6.

2 Related Work

With the rapid development of IoV, the study on VEC has become one of the hot study topics in recent years. Some scholars have done a lot of researches on VEC from the following aspects.

The allocation of communication resources and computing resources in VEC is very important. Wang et al. proposed a multilayer data flow processing system to integrally utilize the computing capacity throughout the whole work, and efficient data processing can be achieved in this way [28]. Liu et al. explored a vehicle edge computing network architecture to maximize the long-term utility of the vehicle edge computing network [29]. Some other scholars also used Deep Reinforcement Learning (DRL) to solve the problems in VEC. Li et al. developed a collaborative edge computing framework to reduce the computing service latency and improve service reliability for vehicular networks, and the offloading and computing problem were formulated as a Markov decision process [30]. Ke et al. designed a task computation offloading model in a heterogeneous vehicular network, and they proposed an adaptive computation offloading method based on DRL to obtain the tradeoff between the cost of energy consumption and the cost of data transmission delay [31]. Load balancing is one of the focuses of VEC. Zhang et al. introduced Fiber-Wireless (FiWi) technology to enhance Vehicular Edge Computing Networks (VECNs), and they proposed a Software-Defined Networking (SDN) based load-balancing task offloading scheme in FiWi enhanced VECNs [32]. Dai et al. proposed integrating load balancing with offloading and studied resource allocation for a multiuser multiserver VEC system. They formulated the joint load balancing and offloading problem as a mixed integer nonlinear programming problem to maximize system utility [33].

The combination of NOMA and VEC has become a new hot research topic. Qian et al. investigated NOMA assisted vehicular edge computing via underlay spectrum sharing. They optimized the vehicular computing-users partial offloading and the allocation of the communication and computing resources to minimize the delay [27]. Zhu et al. constructed a decentralized DRL framework to formulate the power allocation optimization problem due to uncertain multi-input multi-out NOMA channel and stochastic tasks. They adopted the deep deterministic policy gradient algorithm to learn the optimal power allocation scheme based on the decentralized DRL framework [25]. Qian et al. formulated a joint optimization of the computation resource allocations and radio resource

allocations for NOMA transmission, with the objective of minimizing a system wise cost [34].

The above [25–34] have done a great deal of work on vehicle networks from various aspects, and our focus is mainly on the aspect of NOMA-assisted VEC. Although some scholars have conducted research on computational offloading of NOMA-assisted VEC, few scholars have paid attention to the transmission waiting time in the NOMA-transmission process and the selection of edge nodes in the process of vehicle movement, etc. In this paper, we mainly study the transmission batch problem of NOMA and the selection of edge nodes.

3 System Model and Problem Formulation

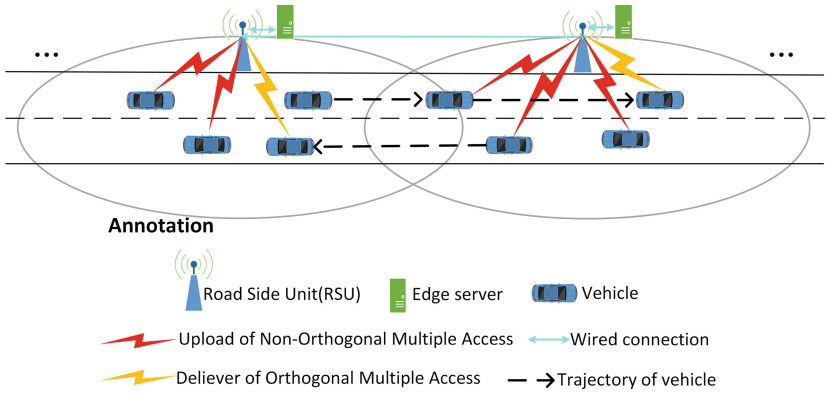

Fig. 1. Network model

Consider the number of equidistant Road Side Units (RSUs) deployed along a straight road, as shown in Fig. 1. Suppose these RSUs are equipped with edge servers, which means they can be used for transmitting and computing simultaneously. In the following we refer to RSUs as edge nodes. Consider vehicles passing the road are generating tasks needed to be handled by these edge nodes, we use the NOMA communication technique for uploading tasks, and NOMA can eliminate the collision between signals from different vehicles. Assume that each task can only choose one edge node for transmission. For one edge node, it may have many tasks to be handled at a time, so it may send some tasks to other nodes for computing. Additionally, we divide the whole schedule time T into multiple identical time slots $t(t \in \mathcal{T})$, where $\mathcal{T}$ is the set of time slots, and we normalize the length of these time slots. Due to the small length of these time slots, each vehicle generates at most one task in each time slot, and tasks generated in the same time slot can be approximately generated at the same time. Because of the fast mobility of vehicles, the number of vehicles on the road

in different time slots will be various, and we define $\mathcal{M}_t$ as the set of vehicles in time slot t.

Define $\mathcal{N}$ as the set of edge nodes. Define $f_{m,t}$ as the task generated by vehicle m at time slot t. The total processing time required for task $f_{m,t}$ can be expressed as follows,

$$T_{m,t}^{Total} = \sum_{x \in \mathcal{N}} \alpha_{m,x,t} \left(T_{m,x,t}^{wait} + T_{m,x,t}^{up} \right) + \sum_{y \in \mathcal{N}} \gamma_{m,y,t} \left(T_{m,y,t}^{que} + T_{m,y,t}^{com} \right), \quad (1)$$

where $T_{m,x,t}^{wait}$ is the waiting time for task $f_{m,t}$ to be uploaded to edge node x, and $T_{m,x,t}^{up}$ is the uploading time for task $f_{m,t}$ to be uploaded to edge node x, and $T_{m,y,t}^{que}$ is the waiting time for task $f_{m,t}$ to be computed on edge node y, and $T_{m,y,t}^{com}$ is the computation time for task $f_{m,t}$ to be computed on edge node y. $\alpha_{m,x,t}$ and $\gamma_{m,y,t}$ are $\{0, 1\}$ variables,

$$\alpha_{m,x,t} = \begin{cases} 1, & \text{if task } f_{m,t} \text{ is transmitted to edge node } x; \\ 0, & \text{otherwise.} \end{cases}$$

$$\gamma_{m,y,t} = \begin{cases} 1, & \text{if task } f_{m,t} \text{ is computed at edge node } y; \\ 0, & \text{otherwise.} \end{cases}$$

Since the data returned by the task is relatively small, we ignore the task return time.

Next, we will introduce the transmission process model and the computation process model respectively.

3.1 Transmission Model

Since we use the NOMA technique, several tasks may be received by one edge node simultaneously. However, the possibility of simultaneous reception is restricted by SINR. So in the transmission model, we combine tasks which can be transmitted together into a batch, and let tasks which cannot be transmitted together join different batches. Then some batches may wait for other batches finishing transmission. As an example shown in Fig. 2. We can see that the second batch is combined with four tasks, and it should wait for the transmitted completion of the first batch. Therefore, we define a $\{0, 1\}$ variable $\theta_{m,n,t}^{j}$ to indicate whether the task $f_{m,t}$ starts to upload in the j-th batch.

$$\theta_{m,n,t}^{j} = \begin{cases} 1, & \text{if } \alpha_{m,n,t} = 1 \text{ and task } f_{m,t} \text{ is uploaded in the } j\text{-th batch;} \\ 0, & \text{otherwise.} \end{cases}$$

So if task $f_{m,t}$ is uploaded in the j-th batch, $T_{m,n,t}^{wait}$ can be expressed as follows,

$$T_{m,n,t}^{wait} = \max_{\forall j | \theta_{m,n,t}^{j} = 1} \left\{ t_n^{j-1} - t, 0 \right\}, \quad (2)$$

where t_n^{j-1} is the time slot for all tasks in the $(j-1)$-th batch to complete the transmission, and it can be expressed as the following formula,

$$t_n^{j-1} = \max_{\forall f_{x,y} | x \in \mathcal{M}_y, y \in \mathcal{T}} \theta_{x,n,y}^{j-1} \left(y + T_{x,n,y}^{wait} + T_{x,n,y}^{up} \right). \tag{3}$$

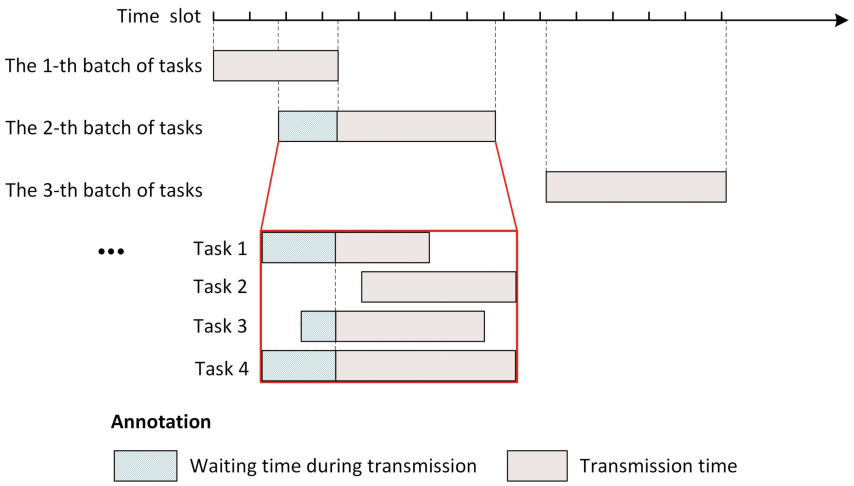

Fig. 2. An example of transmission process

For the convenience of consideration, we approximate the channel gain of vehicle m during the entire transmission process to a constant value, and SINR during the transmission can be expressed as follows,

$$SINR_{m,n,t} = \frac{\theta_{m,n,t}^{j} p g_{m,n,t}}{N_0 + \sum_{\theta_{x,n,y}^{j} p g_{x,n,y} < \theta_{m,n,t}^{j} p g_{m,n,t}} \theta_{x,n,y}^{j} p g_{x,n,y}} \geq \theta_{m,n,t}^{j} \beta, \tag{4}$$

where p is the transmission power of the vehicle, and we suppose all vehicles have the same transmission power. In addition, N_0 is Gaussian white noise, and β is the custom threshold that SINR needs to meet. Moreover, $g_{m,n,t}$ is the channel gain of task $f_{m,t}$, and it can be expressed as follows,

$$g_{m,n,t} = |d_n - d_{m,t}|^{-\lambda}, \tag{5}$$

where $d_{m,t}$ is the position where vehicle m generates task $f_{m,t}$, and d_n is the position of edge node n, and λ is the path loss coefficient [19].

Therefore, The transmission rate $R_{m,n,t}$ of the task $f_{m,t}$ transmitted to the edge node n can be expressed as follows,

$$R_{m,n,t} = W \log_2 \left(1 + SINR_{m,n,t} \right), \tag{6}$$

where W is the uplink bandwidth. According to the above content, we can get the transmission time $T^{up}_{m,n,t}$ of the task $f_{m,t}$,

$$T^{up}_{m,n,t} = \frac{D_{m,t}}{R_{m,n,t}} + T^{RSU}, \tag{7}$$

where $D_{m,t}$ is the amount of data that needs to be transmitted for task $f_{m,t}$, and T^{RSU} is the transmission time between edge nodes. Moreover, since edge nodes are connected by wire, the transmission time between them is quick enough to be ignored. Therefore, we do not consider the T^{RSU} in our system model.

3.2 Computation Model

After the task is transmitted to the edge node n, we will select the appropriate edge node to compute according to the task situation in each edge node and the location of the vehicle. Moreover, we consider edge nodes to compute only one task at a time, so there is a queue waiting time in the computation process. The time slot for task $f_{m,t}$ to reach edge node n can be expressed as follows,

$$a_{m,n,t} = t + T^{wait}_{m,n,t} + T^{up}_{m,n,t}. \tag{8}$$

For tasks arriving at the same time in the time slot $a_{m,n,t}$, they are represented by the set $\mathcal{F}(a_{m,n,t})$. We define a $\{0,1\}$ variable $\rho^{j}_{m,n,t}$, and it indicates whether the task $f_{m,t}$ is computed as the j-th task,

$$\rho^{j}_{m,n,t} = \begin{cases} 1, & \text{if } \gamma_{m,n,t} = 1 \text{ and task } f_{m,t} \text{ is computed as the } j\text{-th task;} \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, if the task $f_{m,t}$ is computed as the j-th task, $T^{que}_{m,n,t}$ is the time required for the $(j-1)$-th task to complete the remaining computation portion at the time slot t. Therefore, the waiting time of the task during the computation process can be expressed as follows,

$$T^{que}_{m,n,t} = \max_{\forall j | \rho^{j}_{m,n,t}=1} \left\{ a^{j-1}_{n} - a_{m,n,t}, 0 \right\}, \tag{9}$$

where a^{j-1}_{n} is the time slot for the $(j-1)$-th task to complete the computation, and it can be expressed as the following formula,

$$a^{j-1}_{n} = \sum_{i=1}^{a_{m,n,t}} \sum_{f_{x,y} \in \mathcal{F}(i)} \rho^{j-1}_{x,n,y} \left(a_{x,n,y} + T^{que}_{x,n,y} + T^{com}_{x,n,y} \right). \tag{10}$$

Finally, the computing time $T^{com}_{m,n,t}$ of task $f_{m,t}$ at edge node n can be obtained as follows,

$$T^{com}_{m,n,t} = \frac{Q_{m,t}}{C}, \tag{11}$$

where $Q_{m,t}$ is the number of CPU cycles required to complete the computation of task $f_{m,t}$, and C is the computing capability of the edge node, and each edge node has the same computing capability.

3.3 Problem Formulation

After a task completes the computation, we need to determine whether the current task is successfully executed. Therefore, we define a $\{0, 1\}$ variable $\mu_{m,t}$, and it indicates whether the task $f_{m,t}$ can be completed within the delay constraint.

$$\mu_{m,t} = \begin{cases} 1, & \text{if}, T_{m,t}^{Total} \leq T_{m,t}^{max}; \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

In addition, $T_{m,t}^{max}$ is maximum delay constraint of task $f_{m,t}$.

The main objective of this paper is to maximize the number of tasks completed by all vehicles in time period T through allocation decisions. And we have the optimization problem as follows,

$$\max \sum_{t \in \mathcal{T}} \sum_{m \in \mathcal{M}_t} \mu_{m,t}, \tag{13}$$

s.t.

$$(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12). \tag{13a}$$

$$\sum_{n \in \mathcal{N}} \alpha_{m,n,t} = 1, \sum_{n \in \mathcal{N}} \gamma_{m,n,t} = 1. \quad \forall m, t \tag{13b}$$

$$\sum_{j} \theta_{m,n,t}^{j} = 1. \quad \forall m, n, t \tag{13c}$$

$$\sum_{t \in \mathcal{T}} \sum_{m \in \mathcal{M}_t} \sum_{j} \rho_{m,n,t}^{j} = 1. \quad \forall n \tag{13d}$$

(13a) indicates the architecture of the entire model. (13b) indicates that each task can only select one edge node in transmission or computation process. (13c) indicates that each task can only select one batch to start uploading. (13d) indicates that each task can only select one order to start computing.

4 Problem Analysis and Algorithm

In this section, we first analyze the optimization problem. In our system model, $\alpha_{m,n,t}$ and $\theta_{m,n,t}^{j}$ are optimization variables in the transmission process, and $\gamma_{m,n,t}$ and $\rho_{m,n,t}^{j}$ are optimization variables in the computation process. According to formula (1), (2), (3), it can be known that $\alpha_{m,n,t}$ and $\theta_{m,n,t}^{j}$ are multiplied. Moreover, $\alpha_{m,n,t}$ and $\theta_{m,n,t}^{j}$ are $\{0, 1\}$ variables. The relationship between $\gamma_{m,n,t}$ and $\rho_{m,n,t}^{j}$ is similar to $\alpha_{m,n,t}$ and $\theta_{m,n,t}^{j}$. Therefore, this is an integer nonlinear optimization problem. This problem is difficult to solve directly, and we consider solving for these variables by heuristics.

Through further analysis, it can be found that the interaction between transmission process and computation process is small, so we choose to solve the optimization variables in these two processes separately. As shown in Algorithm 1, we

Algorithm 1. Asymptotic Inference Greedy Strategy(AIGS) algorithm

Input: $\mathcal{T}$: The set of time slots; $\mathcal{N}$: The set of edge nodes; $\mathcal{M}_t$: The set of vehicle at time slot t; $\mathcal{F}_t$: The set of tasks waiting for transmission in slot t; $\mathcal{F}(t)$: The set of tasks waiting for computation in slot t.

Output: $\alpha_{m,n,t}$, $\theta^j_{m,n,t}$, $\gamma_{m,n,t}$, $\rho^j_{m,n,t}$.

for $t \in \mathcal{T}$ **do**
  When task $f_{m,t}$ is generated in time slot t, $f_{m,t} \rightarrow \mathcal{F}_t$.
  For any task $f_{x,y}$, if $f_{x,y}$ is being uploaded, release $f_{x,y}$ from $\mathcal{F}_t$. If $f_{x,y}$ has completed transmission, $f_{x,y} \rightarrow \mathcal{F}(t)$. If $f_{x,y}$ is being computed, release $f_{x,y}$ from $\mathcal{F}(t)$.
  for $f_{m,t} \in \mathcal{F}_t$ **do**
    In transmission process, get $\alpha_{m,n,t}$ and $\theta^j_{m,n,t}$ according to TPUD algorithm.
  end for
  for $f_{m,t} \in \mathcal{F}(t)$ **do**
    In compute process, get $\gamma_{m,n,t}$ and $\rho^j_{m,n,t}$ according to CPED algorithm.
  end for
end for

design the Asymptotic Inference Greedy Strategy (AIGS) algorithm to solve the optimization problem. We first determine $\alpha_{m,n,t}$ and $\theta^j_{m,n,t}$ in transmission process, and then we solve $\gamma_{m,n,t}$ and $\rho^j_{m,n,t}$ in computation process. In addition, the Transmission Process Upload Decision (TPUD) algorithm and Compute Process Execute Decision (CPED) algorithm will be described in detail in the following sections.

4.1 Problem Analysis in Transmission Process

In the transmission step, we design the TPUD algorithm to solve $\alpha_{m,n,t}$ and $\theta^j_{m,n,t}$. When vehicles generate tasks, edge nodes for uploading tasks should be determined first. We use the following steps for solving $\alpha_{m,n,t}$ and $\theta^j_{m,n,t}$. First, we get an initial method without considering the vehicle's speed. Then, we try to consider the influence of vehicle's speeds from the largest to the smallest. Finally, we try to adjust the initial method until we find an acceptable method.

Step 1: Initialize $\alpha_{m,n,t}$. First, we initialize $\alpha_{m,n,t}$ according to the initial position of the vehicle m. If $d^{min}_n < d_{m,t}$ and $d^{min}_{n+1} > d_{m,t}$, we have $\alpha_{m,n,t} = 1$. Otherwise, we have $\alpha_{m,n,t} = 0$. Moreover, d^{min}_n is the minimum coverage position of edge node n.

Step 2: Get $\theta^j_{m,n,t}$ according to initial $\alpha_{m,n,t}$. Second, we sort tasks according to channel gains from small to large, and use superposition coding to divide these tasks into different batches. Then all $\theta^j_{m,n,t}$ can be obtained.

Step 3: Adjust $\alpha_{m,n,t}$. Third, we narrow the adjustment range of $\alpha_{m,n,t}$, and let tasks only be uploaded to these edge nodes where vehicles can arrive within $T^{max}_{m,t}$. For task $f_{m,t}$ that fails to transmit due to v_m and $d_{m,t}$, we will upload the task to the next edge node and adjust $\alpha_{m,n,t}$.

Algorithm 2. Transmission Process Upload Decision(TPUD) algorithm

Input: $\mathcal{T}$: The set of time slots; $\mathcal{N}$: The set of edge nodes; $\mathcal{M}_t$: The set of vehicle at time slot t; W: Uplink bandwidth; S: Boolean variables used to judge decisions.
Output: $\alpha_{m,n,t}$, $\theta^j_{m,n,t}$.

for $n \in \mathcal{N}$ **do**
  Get the initial $\alpha_{m,n,t}$ according to $d^{min}_n < d_{m,t}$ and $d^{min}_{n+1} > d_{m,t}$.
end for
while $!S$ **do**
  S = TRUE.
  for $n \in \mathcal{N}$ **do**
    Update $\theta^j_{m,n,t}$ according to $SINR_{m,n,t}$.
    if $\left((T^{wait}_{m,n,t} + T^{up}_{m,n,t}) > \frac{d^{min}_{n+1} - d_{m,t}}{v_m}\right)$ **then**
      Adjust $\alpha_{m,n,t}$ and S = FALSE.
    end if
  end for
end while

Step 4: Repeat Step 2 and Step 3. Finally, we will re-solve $\theta^j_{m,n,t}$ according to step 2 and step 3 until all tasks are uploaded successfully or $\alpha_{m,n,t}$ cannot adjust.

4.2 Problem Analysis in Computation Process

In the computation step, we design the CPED algorithm to solve $\gamma_{m,n,t}$ and $\rho^j_{m,n,t}$. After the task is uploaded to the edge node, the appropriate edge node should be selected for computing first. We use the following steps for solving $\gamma_{m,n,t}$ and $\rho^j_{m,n,t}$. First, we get an initial method without considering the cooperation of edge nodes, Then, we try to consider the influence of the cooperation of edge nodes. Finally, we try to adjust the initial method until we find an acceptable method.

Step 1: Initialize $\gamma_{m,n,t}$. First, we initialize $\gamma_{m,n,t}$ according to the $\alpha_{m,n,t}$. Specifically, the task is preferentially computed on the uploaded edge node. If $\alpha_{m,n,t} = 1$, we have $\gamma_{m,n,t} = 1$.

Step 2: Get $\rho^j_{m,n,t}$ according to initial $\gamma_{m,n,t}$. Second, we determine the computation order of each task according to the strategy of high response ratio. Specifically, by comparing $\frac{T^{que}_{m,n,t} + T^{com}_{m,n,t}}{T^{com}_{m,n,t}}$, the task with a highest response ratio will be computed first. Then all $\rho^j_{m,n,t}$ can be obtained.

Step 3: Adjust $\gamma_{m,n,t}$. Third, we narrow the adjustment range of $\gamma_{m,n,t}$, and let tasks only be computed on these edge nodes where vehicles can arrive within $T^{max}_{m,t}$. For task $f_{m,t}$ that fails to compute, if the vehicle m can leave the current edge node within $T^{max}_{m,t}$, we will put the task on the next edge node for computing and adjust $\alpha_{m,n,t}$.

Algorithm 3. Compute Process Execute Decision(CPED) algorithm

Input: $\mathcal{T}$: The set of time slots; $\mathcal{N}$: The set of edge nodes; $\mathcal{M}_t$: The set of vehicle at time slot t; S: Boolean variables used to judge decisions.
Output: $\gamma_{m,n,t}$, $\rho^j_{m,n,t}$.

for $n \in \mathcal{N}$ **do**
  Get the initial $\gamma_{m,n,t}$ according to $\alpha_{m,n,t}$.
end for
while !S **do**
  S = TRUE.
  for $n \in \mathcal{N}$ **do**
    Update $\rho^j_{m,n,t}$ according to $\frac{T^{que}_{m,n,t}+T^{com}_{m,n,t}}{T^{com}_{m,n,t}}$.
    if $\left(T^{Total}_{m,n,t} > T^{max}_{m,t}\right)$ and $\left(\left(d_{m,t} + v_m T^{max}_{m,t}\right) > d^{min}_{n+1}\right)$ **then**
      Adjust $\gamma_{m,n,t}$ and S = FALSE.
    end if
  end for
end while

Step 4: Repeat Step 2 and Step 3. Finally, we will re-solve $\rho^j_{m,n,t}$ according to step 2 and step 3 until all tasks are computed successfully or $\gamma_{m,n,t}$ cannot adjust.

5 Simulation and Experiment

In this section, we mainly introduce the related works of experiments and simulations. We assume that the simulation scene is two bidirectional intersection roads, and RSUs are deployed at roadside intervals of 200m, with each covering a radius of 250 m [30]. Moreover, there are 10 edge nodes serving in our simulation experiments, and these edge nodes communicate over wired connections. We assume that there are 100 vehicles on these roads, and when a vehicle on the road generates a task, it will request nearby edge nodes to assist with the task. And we use a random distribution method to initialize the positions of these vehicles on the two roads. After investigating some real road conditions and experimental data from other papers, we adopt the experimental data in Table 1 to conduct our simulation experiments.

First, we assume that the speed of vehicles is randomly distributed within the interval $[40, 60]$ km/h, and the transmit power of the vehicle is 100 mW. Then, we assume that the threshold for SINR is 1, and the gaussian white noise is -100 dBm [29,34]. Finally, we assume that the size of each task is random, and the data size for task falls within the interval $[5, 10]$Mbits, and the number of required CPU cycles for task falls within the interval $[1000, 3000]$Megacycles, and the maximum delay constraint of task falls within the interval $[1, 3]$s [29,30]. Moreover, we mainly verify the performance of our algorithm by changing the number of tasks requested to be processed, communication bandwidth and edge node computing capability. We simulate the processing of tasks on the road in 30 time slots, and each time slot is 0.2 s long. Since the size of the task is

Table 1. Simulation parameters

Simulation parameters	Value
The transmission power of the vehicle	100 mW
Speed of vehicle	[40, 60]km/h
The custom threshold for SINR	1
Gaussian white noise	−100 dBm
Uplink bandwidth	[5, 10]MHz
The computing capability of the edge node	[5, 10]GHz
Maximum delay constraint of task	[1, 3]s
Data size for task	[5, 10]Mbits
The number of required CPU cycles for task	[1000, 3000]Megacycles

randomly generated, our experimental results are averaged from 50 groups of corresponding experiments.

In our simulation experiments, there are three comparison algorithms. The OMA-AIGS algorithm is obtained by modifying the AIGS algorithm, and the communication method is Orthogonal Multiple Access (OMA). By comparing these two algorithms, we can observe the superiority of NOMA. Moreover, the idea of Random algorithm is that the batches selected during transmission process and the order selected during computation process are random, and the strategy adopted by the FCFS algorithm is that the first come first service.

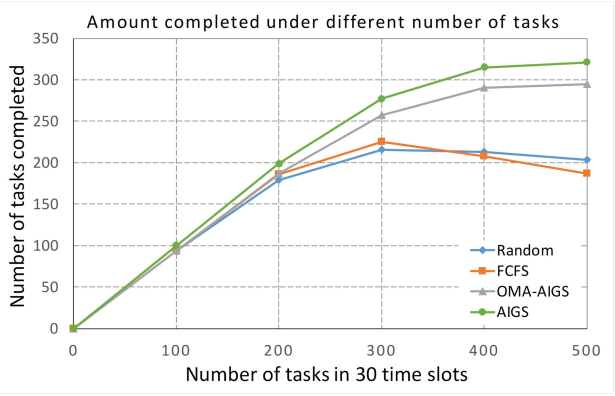

Fig. 3. Different number of tasks

In the first simulation experiment, we believe that the number of tasks requested to be processed has an important impact on algorithm performance. Therefore, we compare the performance of each algorithm under different number

of tasks, and the number of tasks falls within the interval [100, 500]. Moreover, we assume that the uplink bandwidth is 7.5 MHz and the computing capability of the edge node is 7.5 GHz, and our measure of algorithm performance is the number of tasks completed. As shown in Fig. 3, we can know that when the number of tasks requested to be processed increases, the performance of AIGS algorithms is best. Through further observation, we can find that when the number of tasks is large, the performance of Random algorithm and FCFS algorithm begin to degrade, and our algorithm can still remain stable.

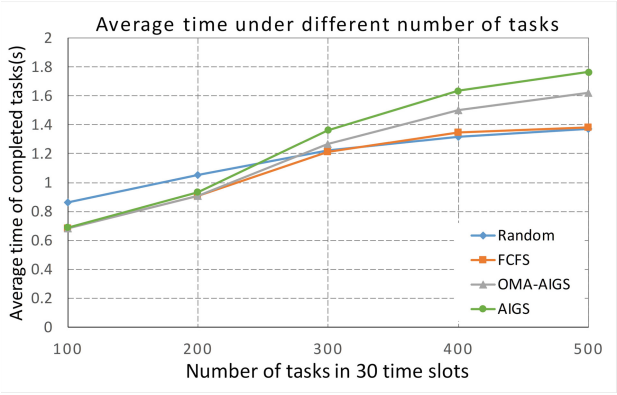

Fig. 4. Average time for complete tasks

In addition, we also compare the average processing time under different number of tasks for these algorithms. As can be seen from Fig. 4, when the number of tasks increases, although the average processing time of our algorithm is longer, the gap with other algorithms is not large, and we are able to accomplish more tasks in this situation. Through the above experimental results, we can know that our algorithm has better performance under different number of tasks.

In the second simulation experiment, we compare the effect of different bandwidth conditions on the performance of the algorithm. We assume that the uplink bandwidth falls within the interval [5, 10] MHz, and the computing capability of the edge node is 7.5 GHz, and the number of tasks is 300. The measure of algorithm performance is the number of tasks completed. As shown in Fig. 5, the number of tasks completed increases gradually for all algorithms as the bandwidth increases. But in general, our algorithm has better performance compared with other algorithms under different bandwidths. In addition, by comparing AIGS algorithm and OMA-AIGS algorithm, we can find that the NOMA method can effectively increase the number of tasks completed.

In the previous simulation experiment, we compare the impact of different bandwidth on the algorithm performance, so in this simulation experiment we compare the impact of different computing capabilities of edge nodes on the

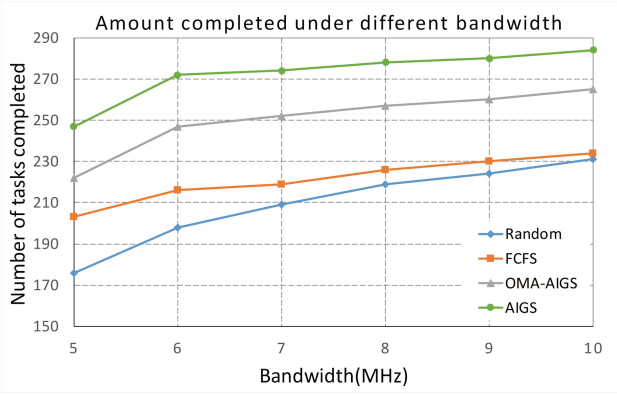

Fig. 5. Different bandwidth

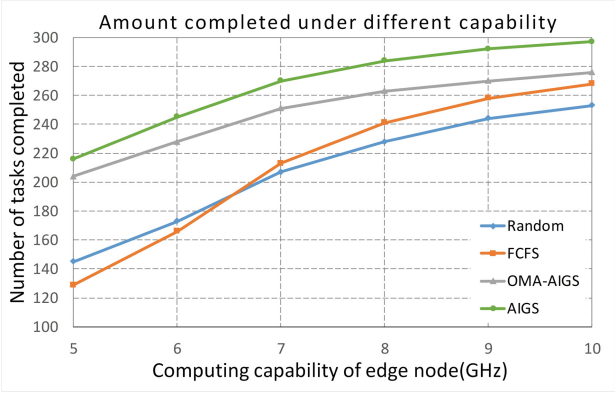

Fig. 6. Different computing capability

algorithm performance. In the final simulation experiment, we assume that computing capability of the edge node falls within the interval $[5, 10]$GHz, and the uplink bandwidth is 7.5 MHz, and the number of tasks is 300. As can be seen from Fig. 6, with the increase of computing power of edge nodes, the number of tasks completed by all algorithms increases gradually, and the overall performance of our algorithm has been the best. Moreover, compared with the impact of different bandwidths on algorithm performance, the impact of different computing power is more obvious.

6 Conclusion

In this paper, we have investigated the task offloading and allocation in IoV. We adopt NOMA-assisted VEC to solve this problem, and divide task processing into two processes. In the transmission process, the transmission method of

NOMA is determined to upload tasks, and according to the position and speed of vehicles, we select appropriate transmission batches for tasks to improve communication quality. In the computation process, the appropriate edge nodes are selected to compute the task according to the mobility of the vehicle. And we have designed AIGS algorithms based on heuristics to solve our problems. We have also presented abundant simulation results to demonstrate the algorithms, and simulation results have shown that our proposed method has effective performance. The main problem in the current research is that the connection between the transmission process and the computation process is ignored, which may not achieve the best results. Therefore, we will make further optimization on the basis of considering the connection between the two processes in future work, and the energy consumption will also be a factor to consider in our optimization problem.

References

1. Kiani, A., Ansari, N.: Edge computing aware NOMA for 5G networks. IEEE Internet Things J. **5**(2), 1299–1306 (2018)
2. Zhang, J., Letaief, K.B.: Mobile edge intelligence and computing for the internet of vehicles. Proc. IEEE **108**(2), 246–261 (2020)
3. Qian, L., Wu, Y., Jiang, F., Yu, N., Lu, W., Lin, B.: NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things. IEEE Trans. Industr. Inf. **17**(8), 5688–5698 (2021)
4. Ding, Z., Fan, P., Poor, H.V.: Impact of non-orthogonal multiple access on the offloading of mobile edge computing. IEEE Trans. Commun. **67**(1), 375–390 (2019)
5. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutorials **19**(4), 2322–2358, Fourthquarter (2017)
6. Saito, Y., Kishiyama, Y., Benjebbour, A., Nakamura, T., Li, A., Higuchi, K.: Non-orthogonal multiple access (NOMA) for cellular future radio access. In: 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), pp. 1–5 (2013)
7. Baghani, M., Parsaeefard, S., Derakhshani, M., Saad, W.: Dynamic non-orthogonal multiple access and orthogonal multiple access in 5G wireless networks. IEEE Trans. Commun. **67**(9), 6360–6373 (2019)
8. Saleem, U., Liu, Y., Jangsher, S., Tao, X., Li, Y.: Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. IEEE Trans. Veh. Technol. **69**(4), 4472–4486 (2020)
9. Liu, Y., Peng, M., Shou, G., Chen, Y., Chen, S.: Toward edge intelligence: multiaccess edge computing for 5G and Internet of Things. IEEE Internet Things J. **7**(8), 6722–6747 (2020)
10. Cui, G., et al.: Demand response in NOMA-based mobile edge computing: a two-phase game-theoretical approach. IEEE Trans. Mobile Comput. (2021)
11. Qian, L., Wu, Y., Ouyang, J., Shi, Z., Lin, B., Jia, W.: Latency optimization for cellular assisted mobile edge computing via non-orthogonal multiple access. IEEE Trans. Veh. Technol. **69**(5), 5494–5507 (2020)
12. Zhang, L., et al.: Energy-efficient non-orthogonal multiple access for downlink communication in mobile edge computing systems. IEEE Trans. Mobile Comput. **21**(12), 4310–4322 (2022). https://doi.org/10.1109/TMC.2021.3083660

13. Hossain, M.A., Ansari, N.: Network slicing for NOMA-enabled edge computing. IEEE Trans. Cloud Comput. (2021)
14. Cui, G., et al.: OL-EUA: online user allocation for NOMA-based mobile edge computing. IEEE Trans. Mobile Comput. (2021)
15. Yang, Z., Liu, Y., Chen, Y., Al-Dhahir, N.: Cache-aided NOMA mobile edge computing: a reinforcement learning approach. IEEE Trans. Wireless Commun. **19**(10), 6899–6915 (2020)
16. Liu, Y.: Exploiting NOMA for cooperative edge computing. IEEE Wirel. Commun. **26**(5), 99–103 (2019)
17. Qian, L., Wu, W., Lu, W., Wu, Y., Lin, B., Quek, T.Q.S.: Secrecy-based energy-efficient mobile edge computing via cooperative non-orthogonal multiple access transmission. IEEE Trans. Commun. **69**(7), 4659–4677 (2021)
18. Tuong, V.D., Truong, T.P., Nguyen, T.-V., Noh, W., Cho, S.: Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning. IEEE Internet Things J. **8**(17), 13196–13208 (2021)
19. Yang, L., Guo, S., Yi, L., Wang, Q., Yang, Y.: NOSCM: a novel offloading strategy for NOMA-enabled hierarchical small cell mobile-edge computing. IEEE Internet Things J. **8**(10), 8107–8118 (2021)
20. Du, J., et al.: When Mobile-Edge Computing (MEC) meets nonorthogonal multiple access (NOMA) for the Internet of Things (IoT): system design and optimization. IEEE Internet Things J. **8**(10), 7849–7862 (2021)
21. Pei, X., Yu, H., Wang, X., Chen, Y., Wen, M., Wu, Y.-C.: NOMA-based pervasive edge computing: secure power allocation for IoV. IEEE Trans. Industr. Inf. **17**(7), 5021–5030 (2021)
22. Tuong, V.D., Noh, W., Cho, S.: Delay minimization for NOMA-enabled mobile edge computing in industrial Internet of Things. IEEE Trans. Industr. Inf. **18**(10), 7321–7331 (2022)
23. Feng, W., et al.: Hybrid beamforming design and resource allocation for UAV-aided wireless-powered mobile edge computing networks with NOMA. IEEE J. Sel. Areas Commun. **39**(11), 3271–3286 (2021)
24. Zhang, X., Zhang, J., Xiong, J., Zhou, L., Wei, J.: Energy-efficient multi-UAV-enabled multiaccess edge computing incorporating NOMA. IEEE Internet Things J. **7**(6), 5613–5627 (2020)
25. Zhu, H., Wu, Q., Wu, X.-J., Fan, Q., Fan, P., Wang, J.: Decentralized power allocation for MIMO-NOMA vehicular edge computing based on deep reinforcement learning. IEEE Internet Things J. **9**(14), 12770–12782 (2022)
26. Arthurs, P., Gillam, L., Krause, P., Wang, N., Halder, K., Mouzakitis, A.: A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles. IEEE Trans. Intell. Transp. Syst. **23**(7), 6206–6221 (2022)
27. Qian, L., Wu, Y., Yu, N., Jiang, F., Zhou, H., Quek, T.Q.S.: Learning driven NOMA assisted vehicular edge computing via underlay spectrum sharing. IEEE Trans. Veh. Technol. **70**(1), 977–992 (2021)
28. Wang, P., Yao, C., Zheng, Z., Sun, G., Song, L.: Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems. IEEE Internet Things J. **6**(2), 2872–2884 (2019)
29. Liu, Y., Yu, H., Xie, S., Zhang, Y.: Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. IEEE Trans. Veh. Technol. **68**(11), 11158–11168 (2019)
30. Li, M., Gao, J., Zhao, L., Shen, X.: Deep reinforcement learning for collaborative edge computing in vehicular networks. IEEE Trans. Cogn. Commun. Netw. **6**(4), 1122–1135 (2020)

31. Ke, H., Wang, J., Deng, L., Ge, Y., Wang, H.: Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks. IEEE Trans. Veh. Technol. **69**(7), 7916–7929 (2020)
32. Zhang, J., Guo, H., Liu, J., Zhang, Y.: Task offloading in vehicular edge computing networks: a load-balancing solution. IEEE Trans. Veh. Technol. **69**(2), 2092–2104 (2020)
33. Dai, Y., Xu, D., Maharjan, S., Zhang, Y.: Joint load balancing and offloading in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4377–4387 (2019)
34. Qian, L.P., Shi, B., Wu, Y., Sun, B., Tsang, D.H.K.: NOMA-enabled mobile edge computing for Internet of Things via joint communication and computation resource allocations. IEEE Internet Things J. **7**(1), 718–733 (2020)