# Edge Computing and UAV Swarm Cooperative Task Offloading in Vehicular Networks

Xiandong Ma+, Zhou Su++, Qichao Xu+, Bincheng Ying+
+School of Mechatronic Engineering and Automation, Shanghai University, China
++School of Cyber Science and Engineering, Xi'an Jiaotong University, China
Corresponding author: zhousu@ieee.org

2022 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-6654-6749-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/IWCMC55113.2022.9824275

***Abstract*—Recently, unmanned aerial vehicle (UAV) swarm has been advocated to provide diverse data-centric services including data relay, content caching and computing task offloading in vehicular networks due to their flexibility and conveniences. Since only offloading computing tasks to edge computing devices (ECDs) can not meet the real-time demand of vehicles in peak traffic flow, this paper proposes to combine edge computing and UAV swarm for cooperative task offloading in vehicular networks. Specifically, we first design a cooperative task offloading framework that vehicles' computing tasks can be executed locally, offloaded to UAV swarm, or offloaded to ECDs. Then, the selection of offloading strategy is formulated as a mixed integer nonlinear programming problem, the object of which is to maximize the utility of the vehicle. To solve the problem, we further decompose the original problem into two subproblems: minimizing the completion time when offloading to UAV swarm and optimizing the computing resources when offloading to ECD. For offloading to UAV swarm, the computing task will be split into multiple subtasks that are offloaded to different UAVs simultaneously for parallel computing. A Q-learning based iterative algorithm is proposed to minimize the computing task's completion time by equalizing the completion time of its subtasks assigned to each UAV. For offloading to ECDs, a gradient descent algorithm is used to optimally allocate computing resources for offloaded tasks. Extensive simulations are lastly conducted to demonstrate that the proposed scheme can significantly improve the utility of vehicles compared with conventional schemes.**

***Index Terms*—Vehicular networks, UAV swarm, edge computing, task offloading.**

## I. Introduction

With the rapid development of vehicular networks in recent years, vehicles have become smarter and connected more closely. Consequently, various vehicle applications (e.g., entertainments, navigation, augmented/virtual reality applications) have increased significantly [1], [2]. However, a large number of computing resources will be occupied when processing these applications' tasks, resulting in a high delay due to the limited computing resources in vehicles.

To reduce the computing delay, edge computing has been applied to realize computing task offloading. Although edge computing presents significant benefits, it still has some deficiencies [3]–[5]. On one hand, edge computing usually relies on fixed edge servers with limited coverage and low flexibility. On the other hand, during peak traffic periods, edge computing devices (ECDs) are prone to overload and oversaturation. Therefore, a flexible and efficient offloading scheme for vehicles should be investigated.

Fortunately, unmanned aerial vehicle (UAV) swarm has been advocated for providing efficient computing task offloading services for vehicles, by harnessing its advantages including low cost, high mobility, and line-of-sight (LoS) communication. Zhang *et al*. [6] use multiple UAVs to execute computing tasks offloaded from mobile devices. Meanwhile, Zhu *et al*. [7] devise a UAVs-assisted computation offloading paradigm where multiple UAVs exist. Besides, Zhang *et al*. [8] exploit multiple UAVs to provide offloading services for mobile devices. Although the above works can realize the computing task offloading, the load balance among UAVs is not sufficiently considered. In addition, rather than offloading the computing task to the same UAV, the computing task can be split into multiple subtasks and executed simultaneously to enjoy the low latency of parallel computing. Keeping that in view, offloading vehicles' computing tasks to UAV swarm is still an open and vital issue.

In this paper, an edge computing and UAV swarm assisted cooperative task offloading scheme in vehicular networks is studied. Specifically, a cooperative task offloading framework where vehicles' computing tasks can be executed locally, offloaded to UAV swarm, or offloaded to ECDs is first designed. In order to maximize the utility of the vehicle, the selection of offloading strategy is then formulated as a mixed integer nonlinear programming problem. Afterwards, the original problem is decomposed into two subproblems: minimizing the completion time when offloading to UAV swarm and optimizing the computing resources when offloading to ECD. Furthermore, when offloading to UAV swarm, a Q-learning based iterative algorithm is proposed to minimize the task completion time. When offloading to ECD, a gradient descent algorithm is utilized to find the optimal computing resources allocated for offloaded tasks. At last, extensive experimental results demonstrate that the proposed scheme outperforms the conventional schemes in terms of the vehicle's utility.

The remainder of this article is organized below. Section II reviews the related work and the system model is described in Section III. Section IV formulates the problem followed by the solution to the subproblems in Section V. The performance of the proposed scheme is evaluated in Section VI. Finally, Section VII closes this paper with conclusions.

## II. Related Work

Various works have been done to improve the efficiency of task offloading in vehicular networks. He *et al*. [9] de-

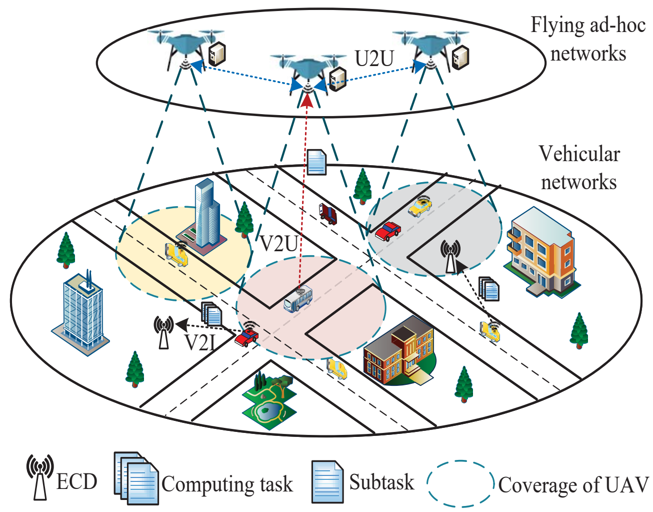


Fig. 1. Network architecture of edge computing and UAV swarm cooperative task offloading in vehicular networks.

signed an edge network that combines caching spaces, edge computational capacities and communication bandwidths with task offloading to enhance vehicles' satisfaction of Quality of Experience (QoE). Misra *et al*. [10] proposed a mobility-aware task offloading scheme, which aims to minimize the task computation time in a software-defined vehicular network. Yadav *et al*. [11] offloaded user tasks to a nearby vehicular node and minimized service latency and energy consumption.

However, the above works only offload vehicles' computing tasks to ECDs, which cannot meet the real-time demand of vehicles in the peak traffic period. In our work, we use edge computing and UAV swarm cooperative task offloading to guarantee vehicles' real-time demand.

Besides, many works use UAVs to help the offloading. Xu *et al*. [12] focused on the computing delay of offloaded tasks executed by multiple UAVs with the aim of minimizing the task completion time. Zhao *et al*. [13] proposed a Software-Defined Networking (SDN) enabled UAV-assisted vehicular computation offloading framework to minimize the system cost of vehicles' computing tasks. Haber *et al*. maximized the rate of served requests by optimizing UAVs' positions, the allocated resources, and the offloading decisions in [14].

However, the above works only offload vehicles' computing tasks to a single UAV, which cannot further reduce the latency. In our work, the computing task is split into several subtasks that are offloaded to different UAVs simultaneously for parallel execution to reduce the completion time.

## III. System Model

### A. Network Model

As shown in Fig. 1, we consider an edge computing and UAV swarm cooperative vehicular network. The flying ad-hoc networks consists of several UAVs that can communicate with each other through UAV-to-UAV (U2U) links. Besides, the vehicular network is established by multiple vehicles and ECDs. There are $N$ UAVs, and the set is denoted by $\mathbf{U} = \{u_1, ..., u_n, ..., u_N\}$. Each vehicle has a computing task to execute and the set of vehicles is denoted as $\mathbf{V} = \{v_1, ..., v_m, ..., v_M\}$, where its total number is $M$. Besides, there exists several ECDs, denoted as $\mathbf{E} = \{e_1, ..., e_s, ...e_S\}$. With Vehicle-to-UAV (V2U) and Vehicle-to-Infrastructure (V2I) links, vehicles can offload their computing tasks to UAV swarm or ECDs.

The task of vehicle $v_m$ is denoted as $T_m$, which can be described by a tuple: $\{D_m, c_m, \tau_m, F_m\}$, where $D_m$ represents the data size (bit), $c_m$ represents the number of CPU cycles needed per bit, $\tau_m$ represents the maximum delay tolerance (s) and $F_m$ represents the total number of CPU cycles required by $T_m$. Apparently, $F_m = c_m D_m$.

Each computing task can be executed with three strategies: local execution, offload to UAV swarm and offload to ECD. Especially, when $v_m$ chooses to offload $T_m$ to UAV swarm for execution, it can be divided into multiple subtasks for parallel computing (described in Sect. V. A). The reason is that subtasks executed in parallel can reduce the completion time of computing tasks. For data-partition oriented applications, the input data is known in advance and can be partitioned arbitrarily for parallel processing because of bit independence. The typical examples are file/graphics compression, virus scanning, visual applications, and object recognition [15], [16].

### B. Communication Model

We consider a three-dimensional coordinate system, the coordinates of vehicle $v_m$, UAV $u_n$ and ECD $e_s$ are $g_{\mathrm{m}} = (x_m(t), y_m(t), 0)$, $q_{\mathrm{n}} = (x_n(t), y_n(t), h_n(t))$ and $o_s = (x_s, y_s, 0)$, respectively. The Orthogonal Frequency Division Multiple Access (OFMDA) technology is adopted in this paper [17]. The bandwidth allocated to each vehicle and each UAV is $B_0$ and $B_1$, respectively. Based on [8], the channel gain of the communication link between $v_m$ and $u_n$ is

$$h_{m,n} = \frac{\beta_0}{||q_n - g_m||^2}, \tag{1}$$

where $\beta_0$ is the channel power gain at the reference distance $d_0 = 1m$. According to Shannon theorem, the communication rate between $v_m$ and $u_n$ is

$$r_{v_m,u_n} = B_0 \log_2(1 + \frac{p_m h_{m,n}}{B_0 N_0}), \tag{2}$$

where $p_m$ is the transmission power of $v_m$ and $N_0$ is the noise power density.

We view the U2U link between UAV $u_n$ and UAV $u_j$ as LoS link, so the communication rate can be calculated by

$$r_{u_n,u_j} = B_1 \log_2(1 + \frac{p_n \beta_1}{||q_n - q_j||^2 B_1 N_0}), \tag{3}$$

where $B_1$ is the bandwidth allocated to each UAV, $\beta_1$ is the channel power gain at the reference distance $d_0 = 1m$.

The communication rate between $v_m$ and $e_s$ is derived as

$$r_{v_m,e_s} = B_0 \log_2(1 + \frac{p_m h^2}{||g_m - o_s||^2 \sigma}), \tag{4}$$

where $h$ is uplink channel fading coefficient and $\sigma$ is Gaussian noise power.

## IV. Problem Formulation

### A. Vehicle's Utility When Computing Locally

The limited local computing resources will result in a long task completion time. The time needed for local computing is

$$t_{T_m,L} = \frac{F_m}{f_m^L}, \tag{5}$$

where $f_m^L$ is the computing resources assigned to $T_m$ from $v_m$. For $v_m$, if $v_m$ obtains short task completion time, $v_m$ will have more utility. Consequently, $v_m$'s utility is

$$s_m^L = C - e^{-(\tau_m - t_{T_m,L})}, \tag{6}$$

where $C$ is a constant which makes the utility non-negative.

### B. Vehicle's Utility When Offloading to UAV Swarm

In pursuit of low latency, $T_m$ can be decomposed into several subtasks, each of them will be executed by different UAVs simultaneously, i.e., $T_m = \{t_{m,1}, ..., t_{m,i}, ..., t_{m,I}\} (I \le N)$ will be offloaded to $U_m = \{u_{m,1}, ..., u_{m,i}, ..., u_{m,I}\}$. The data size set of the subtask is $D_{T_m} = (d_{m,1}, ..., d_{m,i}, ..., d_{m,I})$, where $d_{m,i}$ is $t_{m,i}$'s data size. The total number of CPU cycles required by $t_{m,i}$ is $f_{m,i}$ and $f_{m,i} = c_m d_{m,i}$. According to the above definition, we can obtain the following formula:

$$\sum_{i=1}^{I} d_{m,i} = D_m, \sum_{i=1}^{I} f_{m,i} = F_m, \forall v_m \in \mathbf{V}. \tag{7}$$

When subtask $t_{m,i}$ is offloaded to UAV $u_{m,i}$, $t_{m,i}$ can be transmitted directly if $v_m$ is within $u_{m,i}$'s communication range. Otherwise, $t_{m,i}$ needs a relay UAV. We introduce a binary variable $x_m^i$. $x_m^i = 1$ denotes that $v_m$ is connected with $u_{m,i}$, otherwise $x_m^i = 0$. Since UAVs can communicate with each other, when $x_m^i = 0$, the relay UAV is

$$u_{m,i}^{relay} = \underset{u_j \in \mathbf{U}, u_j \ne u_{m,i}}{\arg} \max x_{m,j} \frac{1}{\frac{1}{r_{v_m,u_j}} + \frac{1}{r_{u_j,u_{m,i}}}}, \tag{8}$$

where $\frac{1}{r_{v_m,u_j}} + \frac{1}{r_{u_j,u_{m,i}}}$ is the time required to transmit one bit of data from $v_m$ to $u_{m,i}$ through $u_j$, i.e., $\frac{1}{r_{v_m,u_{m,i}}}$. Therefore, the transmission time is

$$t_{t_{m,i},u_{m,i}}^{tra} = \frac{d_{m,i}}{r_{v_m,u_{m,i}}}. \tag{9}$$

If subtasks are assigned randomly without considering the load of each UAV, the computing task's completion time will be prolonged. We use $F_{m,i}^{wait}$ to represent the CPU cycles required for the remaining tasks of $u_{m,i}$, then $u_{m,i}$ needs to wait before executing its next subtask. The waiting time is

$$t_{u_{m,i}}^{wait} = \frac{F_{m,i}^{wait}}{f_u}, \tag{10}$$

where $f_u$ is the amount of computing resources allocated from each UAV to computing tasks. With the help of $t_{u_{m,i}}^{wait}$, the current load of $u_{m,i}$ can be obtained. The computing time required for $u_{m,i}$ to execute subtask $t_{m,i}$ is

$$t_{u_{m,i},t_{m,i}}^{cmpt} = \frac{f_{m,i}}{f_u}. \tag{11}$$

Because the data size of computing results is relatively small, we omit the transmit time of results. Since $u_{m,i}$ is still executing the remaining tasks during $t_{m,i}$'s transmission, $t_{m,i}$ can be executed immediately if $t_{t_{m,i},u_{m,i}}^{tra}$ is greater than $t_{u_{m,i}}^{wait}$. Therefore, the total time required to execute subtask $t_{m,i}$ is

$$t_{t_{m,i},u_{m,i}} = \max(t_{t_{m,i},u_{m,i}}^{tra}, t_{u_{m,i}}^{wait}) + t_{u_{m,i},t_{m,i}}^{cmpt}. \tag{12}$$

As $T_m$ is divided into several subtasks, the execution time of $T_m$ is determined by the largest execution time, i.e.,

$$t_{T_m,U} = \{t_{t_{m,i},u_{m,i}} | \forall t_{m,k} \in T_m : t_{t_{m,i},u_{m,i}} \ge t_{t_{m,k},u_{m,k}}\}. \tag{13}$$

As executing offloaded tasks consumes considerable battery energy of UAVs, the vehicle has to pay the UAVs. Similar to [18], the communication energy is ignored since it's too small compared to the computing energy. Therefore, the payment for offloading computing task $T_m$ to UAV swarm is

$$p_{T_m,U} = \sum_{i=1}^{I} \gamma_U \delta d_{m,i} (f_u)^2 = \gamma_U \delta F_m (f_u)^2, \tag{14}$$

where $\gamma_U$ is UAV's unit price of energy and $\delta$ is a constant related to the hardware architecture. We use the satisfaction of the vehicle and the payment to the UAV swarm to measure the utility of the vehicle. Thereby, $v_m$'s utility is

$$s_m^U = \alpha(C - e^{-(\tau_m - t_{T_m,U})}) - p_{T_m,U}, \tag{15}$$

where $\alpha$ is the offloading willingness.

### C. Vehicle's Utility When Offloading to ECD

According to the principle of proximity, the nearest ECD will be selected. The time required for transmitting $T_m$ from $v_m$ to $e_s$ is $t_{T_m,e_s}^{tra} = \frac{D_m}{r_{v_m,e_s}}$. The ECD $e_s$ will compute the task as soon as it arrives, the time of $e_s$ calculating $T_m$ is

$$t_{e_s,T_m}^{cmpt} = \frac{F_m}{f_m^{e_s}}, \tag{16}$$

where $f_m^{e_s}$ is the allocated computing resources from $e_s$ to $T_m$. Therefore, $T_m$'s completion time is

$$t_{T_m,e_s} = t_{T_m,e_s}^{tra} + t_{e_s,T_m}^{cmpt}. \tag{17}$$

Similar to offloading to UAV swarm, the vehicle has to pay the ECD. Denote $\gamma_E$ as the unit price of ECD's energy, then the utility of $v_m$ when offloading to $e_s$ is

$$s_m^E = \alpha(C - e^{-(\tau_m - t_{T_m,e_s})}) - \gamma_E \delta F_m (f_m^{e_s})^2. \tag{18}$$

### D. Problem Formulation

We introduce $d_m = \{d_m^L, d_m^U, d_m^E\}$ to describe the computing strategy. Binary variable $d_m^L$, $d_m^U$, $d_m^E$ indicate that whether $T_m$ is executed locally, offloaded to UAV swarm, or offloaded to ECD, respectively. $d_m^L = 1$ means that $T_m$ is executed locally, $d_m^L = 0$ means other situations. $d_m^U$ and $d_m^E$ are similar to $d_m^L$. This paper aims to maximize the utility of vehicle, and the optimization problem is formulated as:

$$\begin{aligned} P_1 : & \max_{d_m} (d_m^L s_m^L + d_m^U s_m^U + d_m^E s_m^E) \\ s.t.\ & \mathrm{C}_1 : d_m^L + d_m^U + d_m^E \le 1 \\ & \mathrm{C}_2 : d_m^L t_{T_m,L} + d_m^U t_{T_m,U} + d_m^E t_{T_m,e_s} \le \tau_m \end{aligned}, \tag{19}$$

where $C_1$ indicates that the computing task can only be executed locally, offloaded to UAV swarm, or offloaded to ECD. $C_2$ means that $T_m$'s completion time should be less than the tolerable delay $\tau_m$.

## V. Proposed Optimization Approach

$P_1$ is NP-hard since it is a 0-1 integer programming problem that cannot be solved in linear time. By observing the problem, we decompose it into three subproblems: 1) maximizing the utility when offloading to UAV swarm, 2) maximizing the utility when offloading to ECD, and 3) maximizing the utility when the task is computed locally.

### *A. Maximize Vehicle's Utility When Offloading to UAV swarm*

According to Eq. (15), $s_m^U$ is determined by $t_{T_m,U}$. As long as $t_{T_m,U}$ is minimized, $s_m^U$ is maximized. Therefore, the subproblem of maximizing $s_m^U$ can be transformed into minimizing $t_{T_m,U}$, i.e.,

$$\begin{aligned} P_2 : &\min_{D_{T_m}} t_{T_m,U} \\ &s.t.\ t_{T_m,U} \le \tau_m \end{aligned} . \tag{20}$$

Obviously, the minimum latency $t^{\min}_{T_m,U}$ is reached when the parallel subtasks take the same completion time. Therefore, the first step is to determine UAVs allocation state, i.e., which UAVs are selected to execute $T_m$'s subtasks. As a kind of reinforcement learning, Q-learning is an effective solution when the number of actions and states of a decision problem is limited [19]. Consequently, the Q-learning approach is used to find the optimal UAVs allocation state $s^*$ and the Q-learning can be expressed as a tuple $\{S,A,R(s,a)\}$. Specially,

1) $S = \{s_1, ..., s_i, ..., s_I\}$ is the allocation states set of UAVs, where $s_i = (u_{m,1}, u_{m,2}, ..., u_{m,i})$ represents the set of UAVs allocated to execute $T_m$'s subtasks. To realize load balance, the UAVs in status $s_i$ are selected from $\mathbf{U}$ in ascending order of waiting time, i.e., $t^{wait}_{u_{m,1}} < t^{wait}_{u_{m,2}} < \cdots < t^{wait}_{u_{m,i}}$.

2) $A = \{a_1, ..., a_i, ..., a_I\}$ represents the set of replacement actions for each state, in which $a_i = (u_i^-, u_{i+1}^+)$ represents the replacement action can be taken in state $s_i$. $a_i = (1, 0)$ indicates that $u_{m,i}$ is removed from the allocation set to reach state $s_{i-1}$. $a_i = (0, 1)$ indicates that $u_{m,i+1}$ is added to the allocation set to reach status $s_{i+1}$.

3) $R(s, a) = \{R_1(s_1, a_1), ..., R_i(s_i, a_i), ..., R_I(s_I, a_I)\}$ is the reward set of performing action $a_i$ in state $s_i$.

The Q-value is described as the state-action pair $Q(s_i, a_i)$, and the update of the Q-value can be expressed as:

$$\begin{aligned} Q(s_i, a_i) = (1-\nu)&Q(s_i, a_i) + \\ &\nu[R_i(s_i, a_i) + \gamma \max_{a_i} Q(s_{i'}, a_{i'})], \end{aligned} \tag{21}$$

where $s_i$ is the current allocation state of UAVs. $s_{i'}$ is the new allocation state of UAVs after the execution of action $a_i$, and $a_{i'}$ is the action that can be taken in state $s_{i'}$. $\nu$ is the learning rate and $\gamma$ is the discount factor used to determine the impact of future rewards on present rewards.

**Algorithm 1 : Q-Learning based Iterative Algorithm**

**Initialize** Sort U decrementally by waiting time;
**Repeat**
Selects a UAV replacement action $a_i$;
Calculate the reward according to Eq. (22);
Update Q-value according to Eq. (21);
$s_{i'} \leftarrow s_i$;
$a_{i'} \leftarrow a_i$;
**Until** Q-value converges
Find $s^*$ according to Q-value;
Calculate $t^{\min}_{sub}$, $t^{\max}_{sub}$ in $s^*$ and set $t^{\min}_{T_m,U} = t^{\min}_{sub}$;
**while** $t^{\min}_{T_m,U} < t^{\max}_{sub}$ **do**
    Calculate $d_{m,k}$ in $s^*$ according to Eq. (24) and Eq. (26);
    **if** $\sum_1^{I^*} d_{m,i} - D_m < \varepsilon_1$ **then**
        **Break**
    **end if**
    $t^{\min}_{T_m,U} = t^{\min}_{T_m,U} + \varepsilon_2$;
**end while**
Calculate $s_m^U$ according to Eq. (15);

In order to find the optimal UAVs allocation state and achieve load balance, we formulate $R_i(s_i, a_i)$ as follows:

$$R_i(s_i, a_i) = \frac{F_m}{\sum_{k=1}^{i'} d_{m,k}\Big|_{t_{T_m,U} = t^{wait}_{u_{m,i'+1}}} - F_m}, \tag{22}$$

where the first term in the denominator represents the sum of data size of $T_m$'s subtasks when the completion time of each subtask is $t^{wait}_{u_{m,i'+1}}$ in state $s_{i'}$. The implication of Eq. (22) is that if state $s_i$ can complete task $T_m$ within the time of $t^{wait}_{u_{m,i+1}}$, there is no need to transform from $s_i$ to $s_{i+1}$. The reason is that the time to complete $T_m$ in $s_{i+1}$ must be longer than $t^{wait}_{u_{m,i+1}}$. In this way, the task completion time is reduced and load balance can be ensured. In the next, we inversely calculate $d_{m,k}$ of each subtask assigned to UAVs in state $s_i$ according to the task completion time $t_{T_m,U}$.

In status $s_i$, there are two possible situations for each UAV $u_{m,k} (1 \le k \le i)$ after assigning subtasks: $t^{tra}_{t_{m,k},u_{m,k}} \le t^{wait}_{u_{m,k}}$ or $t^{tra}_{t_{m,k},u_{m,k}} > t^{wait}_{u_{m,k}}$. The boundary condition of these two situations is that $t^{tra}_{t_{m,k},u_{m,k}} = t^{wait}_{u_{m,k}}$. Then, the subtask's data size under this boundary condition is $t^{wait}_{u_{m,k}} r_{v_m u_{m,k}}$, and the total completion time required for this part of data is

$$t^{bor}_{t_{m,k},u_{m,k}} = t^{wait}_{u_{m,k}} + \frac{t^{wait}_{u_{m,k}} r_{v_m u_{m,k}}}{D_m f_u} F_m, \tag{23}$$

where the last item is the calculation time.

Therefore, if $t_{T_m,U} \le t^{bor}_{t_{m,k},u_{m,k}}$, it will be the first case and the data size of the subtask offloaded to $u_{m,k}$ is

$$d_{m,k} = D_m \frac{\left(t_{T_m,U} - t^{wait}_{u_{m,k}}\right) f_u}{F_m}. \tag{24}$$

If $t_{T_m,U} > t^{bor}_{t_{m,k},u_{m,k}}$, it will be the second case, where $d_{m,k}$ can be divided into two parts. The transmitting time of the first part overlaps with the waiting time, so the data size of the first part is $d^f_{m,k} = t^{wait}_{u_{m,k}} r_{v_m u_{m,k}}$, and the total time required for the first part is shown in Eq. (23). Therefore, the total time consumed by the second part is $t_{T_m,U} - t^{bor}_{t_{m,k},u_{m,k}}$. The second part consumes not only additional transmission time, but also computation time. As a result, the data size of the second part is

$$d^s_{m,k} = \frac{t_{T_m,U} - t^{bor}_{t_{m,k},u_{m,k}}}{\frac{1}{r_{v_m,u_{m,k}}} + \frac{1}{D_m}\frac{F_m}{f_u}}, \tag{25}$$

where the denominator is the transmission time and computation time required for one bit of data. As a result, in the second case:

$$d_{m,k} = d^f_{m,k} + d^s_{m,k}. \tag{26}$$

After achieving $d_{m,k}$, the optimal UAVs allocation state $s^*$ can be obtained through Q-learning. The next step is to find each subtask's data size and the match between subtasks and UAVs, i.e., the data size of each subtask assigned to each UAV in $s^*$. First, we set up subtasks of equal data size for each UAV, i.e., $d_{m,i} = \frac{D_m}{I^*}, (1 \le i \le I^*)$. $I^*$ is the number of UAVs in $s^*$. Then, we figure out the real completion time of subtasks according to Eq. (12). The shortest completion time and the longest completion time among them are denoted as $t^{\min}_{sub}$ and $t^{\max}_{sub}$, respectively, i.e., $t^{\min}_{sub} = \min(t_{t_{m,i},u_{m,i}} | 1 \le i \le I^*)$ and $t^{\max}_{sub} = \max(t_{t_{m,i},u_{m,i}} | 1 \le i \le I^*)$.

Because $t^{\min}_{T_m,U}$ must exist between $t^{\min}_{sub}$ and $t^{\max}_{sub}$ and there is not much numerical difference between $t^{\min}_{sub}$ and $t^{\max}_{sub}$, so we use the iterative method to find out $t^{\min}_{T_m,U}$. Obviously, after $t^{\min}_{T_m,U}$ is determined, we can figure out the data size of each subtask like $d_{m,k}$.

The concrete implementation process is shown in Algorithm 1. We first sort UAVs in ascending order of load so that low load UAVs would be preferentially selected to execute subtasks, which is conducive to load balance. In Line 2 to Line 9, Q-learning is used to find the optimal UAVs allocation state $s^*$. Afterwards, the minimum task completion time is obtained by iteration in Line 10 to Line 18. Since the time required for each UAV to complete its respective subtask is equal, the load balance among UAVs is further satisfied.

### B. Maximize Vehicle's Utility When Offloading to ECD

The subproblem of maximizing the vehicle's utility when the computing task offloaded to ECD can be formulated as

$$\begin{aligned} P_3 : & \max_{f^{e_s}_m} s^E_m \\ & s.t. \;\; t_{T_m,e_s} \le \tau_m \end{aligned}. \tag{27}$$

According to Eq. (18), the utility of $v_m$ is determined by $f^{e_s}_m$. We first calculate the first-order partial derivative of $s^E_m$ with respect to $f^{e_s}_m$:

$$\frac{\partial s^E_m}{\partial f^{e_s}_m} = \frac{\alpha F_m}{(f^{e_s}_m)^2} e^{\frac{F_m}{f^{e_s}_m} + t^{tra}_{T_m,e_s} - \tau_m} - 2\gamma_E \delta F_m f^{e_s}_m, \tag{28}$$

Then, we calculate the second-order partial derivative of $s^E_m$ with respect to $f^{e_s}_m$:

$$\frac{\partial^2 s^E_m}{\partial (f^{e_s}_m)^2} = -(2 + \frac{F_m}{f^{e_s}_m}) \frac{\alpha F_m}{(f^{e_s}_m)^3} e^{\frac{F_m}{f^{e_s}_m} + t^{tra}_{T_m,e_s} - \tau_m} - 2\gamma_E \delta F_m. \tag{29}$$

**Algorithm 2 : Gradient Descent Algorithm**

1: **Initialize** $f^{e_s(0)}_m$, $k = 0$, $step$;
2: $s(f^{e_s}_m) = -s^E_m$;
3: **while** $s(f^{e_s(k)}_m) - s(f^{e_s(k+1)}_m) > \varepsilon$ **do**
4: $\quad step = \frac{(\nabla s(f^{e_s(k)}_m))^T \nabla s(f^{e_s(k)}_m)}{\nabla s(f^{e_s(k)}_m))^T \nabla^2 s(f^{e_s(k)}_m) \nabla s(f^{e_s(k)}_m)}$;
5: $\quad f^{e_s(k+1)}_m = f^{e_s(k)}_m - step \times \nabla s(f^{e_s(k)}_m)$;
6: **end while**
7: Calculate $s^E_m$ according to Eq. (18);

Due to the negativity of the second derivative, $s^E_m$ is concave, i.e., there exists the maximum utility for the vehicle. We then conduct the following operations:

$$\lim_{f^{e_s}_m \to 0^+} \frac{\partial s^E_m}{\partial f^{e_s}_m} = +\infty, \lim_{f^{e_s}_m \to +\infty} \frac{\partial s^E_m}{\partial f^{e_s}_m} = -\infty. \tag{30}$$

Therefore, let Eq. (28) be equal to zero, we can get the optimal computing resource $f^{e^*_s}_m$. Since it can not be solved directly, we use the gradient descent algorithm to approximate $f^{e^*_s}_m$ as shown is Algorithm 2. In Line 2, we first find the negative of $s^E_m$ because the maximum value of $s^E_m$ is the minimum value of $-s^E_m$. The gradient descent algorithm is then used to find the optimal computing resource from Line 3 to Line 6, where $\nabla s(f^{e_s(k)}_m) = -\frac{\partial s^E_m}{\partial f^{e_s}_m}$ and $\nabla^2 s(f^{e_s(k)}_m) = -\frac{\partial^2 s^E_m}{\partial (f^{e_s}_m)^2}$.

### C. Maximize Vehicle's Utility When Computing Locally

Since $s^L_m$ is a monotonic function of $f^L_m$, the greater $f^L_m$, the greater $s^L_m$. $s^L_m$ is a fixed value when $f^L_m$ is determined, so we do not need to consider this subproblem.

## VI. Performance Evaluations

### A. Simulation Setup

We consider a 500m×500m simulation area, in which there are 9 uniformly distributed UAVs and 4 uniformly distributed ECDs. Vehicles' coordinates are randomly selected from the area and each vehicle has a computing task to execute. The computing capacity of the vehicle is $f^L_m = 1$GHZ, the computing resource that each UAV can provide is $f_u = 4$GHZ. The CPU cycles required for one bit is $c_m = 1000$ [8] and channel power gain $\beta_0 = \beta_1 = -50$dB at reference distance $d_0 = 1$m. For the computation tasks, the data size ranges from 100KB to 500KB [15] and the flying height of UAV is 20m [17]. Besides, $\delta = 1 \times 10^{-28}$ [18].

Since [6]–[8] offload the computing task to a single UAV, we compare our proposal with the scheme that offloads tasks to a greedily selected UAV (OTU). In OTU, vehicles' computing tasks are offloaded to a UAV selected from a greedy algorithm to ensure the vehicle obtains more utility. Another scheme is computing the task locally (CTL) that vehicles' computing tasks are executed locally.

### B. Performance Evaluation

Fig. 2 shows the vehicle's utility obtains from three schemes when the data size of the computing task changes. For our proposed scheme, subtasks are executed in parallel and the optimal computing resources are used. Consequently, the utility obtained from our proposed scheme is the largest. When the data size of the computing task becomes larger, the limited computing resources of a single UAV cannot complete the computing task in a short time. Therefore, the utility obtained from OTU is becoming less and less.

Fig. 3 shows the vehicle's utility obtained from offloading computing tasks to ECD when the ECD allocates different computing resources. In Fig. 3, there are three schemes to determine $f_m^{e_s}$ that $e_s$ allocates to the offloaded computing task: optimal computing resources $f_m^{e_s*}$ determined by our scheme, $f_m^{e_s} = 4$GHZ and $f_m^{e_s} = 5$GHZ. As seen from Fig. 3, our scheme is always the most effective. The reason is that the vehicle's utility when offloading to the ECD is mainly determined by the computing resources allocated to the computing task.

## VII. Conclusion

In this paper, we have proposed an edge computing and UAV swarm assisted cooperative task offloading scheme in vehicular networks to guarantee vehicles' real-time demand in peak traffic. Specifically, a cooperative task offloading framework is first designed in which vehicles' computing tasks can be executed locally, offloaded to UAV swarm, or offloaded to ECDs. A mixed integer nonlinear programming problem is then formulated to maximize the utility of the vehicle by finding the optimal offloading strategy. Furthermore, the original problem is decomposed into two subproblems: minimizing the completion time when the computing task is offloaded to UAV swarm and optimizing the computing resources when offloaded to ECD. When offloading to UAV swarm, a Q-learning based iterative algorithm that can realize the load balance among UAVs is designed to minimize the completion time. When offloading to ECDs, a gradient descent algorithm is utilized to find the optimal computing resources allocated for computing tasks. At last, numerical simulations have demonstrated the effectiveness and superiority of our proposed scheme in comparison with conventional schemes. In future work, the communication security between vehicles and UAVs will be studied.

## Acknowledgment

This work is supported in part by NSFC (Nos. U1808207, U20A20175, and 62001278), NSF of Shanghai (No. 20ZR1420700), and Shanghai Sailing Program (No. 20YF1413600).

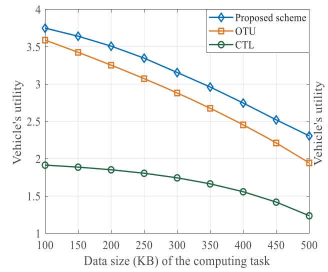


Fig. 2. Utility of the vehicle under different schemes.

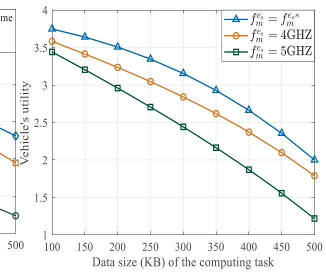


Fig. 3. Utility of the vehicle ubder different $f_m^{e_s}$.

## References

[1] Z. Su, M. Dai, Q. Xu, R. Li and H. Zhang, "UAV Enabled Content Distribution for Internet of Connected Vehicles in 5G Heterogeneous Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5091-5102, Aug. 2021.

[2] Y. Wang *et al*., "Task Offloading for Post-Disaster Rescue in Unmanned Aerial Vehicles Networks," *IEEE/ACM Transactions on Networking*, doi: 10.1109/TNET.2022.3140796.

[3] Y. Hui *et al*., "Collaboration as a Service: Digital Twins Enabled Collaborative and Distributed Autonomous Driving," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2022.3161677.

[4] Z. Wang, D. Zhao, M. Ni, L. Li and C. Li, "Collaborative Mobile Computation Offloading to Vehicle-Based Cloudlets," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 768-781, Jan. 2021.

[5] Q. Xu, Z. Su, R. Lu and S. Yu, "Ubiquitous Transmission Service: Hierarchical Wireless Data Rate Provisioning in Space-Air-Ocean Integrated Networks," *IEEE Transactions on Wireless Communications*, doi: 10.1109/TWC.2022.3162400.

[6] L. Zhang and N. Ansari, "Optimizing the Operation Cost for UAV-Aided Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6085-6093, June 2021.

[7] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang and X. Lang, "Learning-Based Computation Offloading Approaches in UAVs-Assisted Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 928-944, Jan. 2021.

[8] J. Zhang *et al*., "Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2114-2125, Feb. 2020.

[9] X. He, H. Lu, M. Du, Y. Mao and K. Wang, "QoE-Based Task Offloading With Deep Reinforcement Learning in Edge-Enabled Internet of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2252-2261, April 2021.

[10] S. Misra and S. Bera, "Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2071-2078, Feb. 2020.

[11] R. Yadav, W. Zhang, O. Kaiwartya, H. Song and S. Yu, "Energy-Latency Tradeoff for Dynamic Computation Offloading in Vehicular Fog Computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14198-14211, Dec. 2020.

[12] Y. Xu, T. Zhang, J. Loo, D. Yang and L. Xiao, "Completion Time Minimization for UAV-Assisted Mobile-Edge Computing Systems," *IEEE Transactions on Vehicular Technology*, doi: 10.1109/TVT.2021.3112853.

[13] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma and Z. Liu, "A Novel Cost Optimization Strategy for SDN-Enabled UAV-Assisted Vehicular Computation Offloading," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3664-3674, June 2021.

[14] E. El Haber, H. A. Alameddine, C. Assi and S. Sharafeddine, "UAV-aided Ultra-Reliable Low-Latency Computation Offloading in Future IoT Networks," *IEEE Transactions on Communications*, doi: 10.1109/TCOMM.2021.3096559.

[15] U. Saleem, Y. Liu, S. Jangsher, X. Tao and Y. Li, "Latency Minimization for D2D-Enabled Partial Computation Offloading in Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472-4486, April 2020.

[16] Y. Liu, Z. Su and Y. Wang, "Energy-Efficient and Physical Layer Secure Computation Offloading in Blockchain-Empowered Internet of Things," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2022.3159248.

[17] T. Zhang, Y. Xu, J. Loo, D. Yang and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5505-5516, Aug. 2020.

[18] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao and X. Shen, "Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 3, pp. 3424-3438, March 2020.

[19] Z. Su, Y. Wang, Q. Xu and N. Zhang, "LVBS: Lightweight Vehicular Blockchain for Secure Data Sharing in Disaster Rescue," *IEEE Transactions on Dependable and Secure Computing*, doi: 10.1109/TDSC.2020.2980255.