Task Scheduling with UAV-assisted Vehicular Cloud for Road Detection in Highway Scenario

Jiangfan Li, Xiaofeng Cao, Deke Guo*, Junjie Xie, Honghui Chen

Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology, Changsha Hunan 410073, P.R. China

***Abstract*—Vehicular Cloud Computing (VCC) has been utilized to enhance the traffic management and the road safety. By connecting with base stations (BSes), VCC can provide the information of real-time dynamics for smart vehicles (SVs). However, the area outside the coverage of BSes will be the blind areas, where smart vehicles can not obtain the real-time safety guarantee, especially on the highway. In this paper, we utilize Unmanned Aerial Vehicles (UAVs) to assist the communication between SVs and BSes to solve the above problem. In particular, we study inter-dependency tasks scheduling for the highway driving environment detection, where SVs, BSes and UAVs collect the environmental data, schedule tasks and feedback results cooperatively. There are two main problems in this scenario, the scheduling within the coverage of BSes and the rescheduling between the coverage of BSes. We model both the processes as constrained numerical optimization problems aiming to minimize the request response time. To this end, we propose a systematical scheduling scheme named Teso, which consists of two stages: 1) designed approximation algorithm for scheduling; 2) offloading algorithm for rescheduling. Extensive experiments show that Teso can significantly reduce the response time overall and improve the system stability.**

I. Introduction

With the rapid development of autonomous driving [1] [2], the smart vehicles (SVs) with varying levels of automation are already on the road. As the advanced computing and communication capabilities of SVs, it is widely expected that SVs can achieve safer and more efficient transportation. To this end, Vehicular Cloud Computing (VCC) [3] has been proposed to enable the interconnection among nearby SVs for the efficient resource utilization and application executing (e.g., road environment detection and monitoring) [4] [5]. In the VCC, SVs are seen as mobile servers, and can be connected with SVs nearby by vehicular ad-hoc networks (VANETs) [6]. Based on this, the resource of multiple SVs can be pooled together and jointly utilized for various mobile services [7]. One of the important services is driving environment detection [8], which is the essential component of Intelligent Transportation Systems [9].

Although SVs with the rich on-board sensors (e.g., Radar, camera) can perceive the surrounding environment, the fast and accurate driving decisions cannot be enabled without pre-acquired road information, especially on the highway. For instance, vehicles are at a high speed and any anomalous factor can cause a fatal accident. Therefore, there should be a monitoring platform to detect anomalies (e.g., a sudden obstacle and an accident just happening) on the highway in real time. As it is not practical to install cameras to cover all roads, the on-board sensors of SVs can be the eye of the platform to collect the real-time anomaly information. Based on such information, the platform can communicate with those SVs on the highway to make some prevention and treatment measures to those anomalies. Therefore, we utilize VCC, consisting of multiple SVs, to detect the road environment and find the anomalies on the highway. Multiple SVs work together can get more comprehensive road information and make monitoring with high precision for the future driving. Even though an SV can detect the road by itself, it is difficult for a single SV to collect complete information about the highway environment. Collaborative SVs can also make some tasks processed in parallel, which can reduce the detection time and avoid occupying the resources of an SV for a long time.

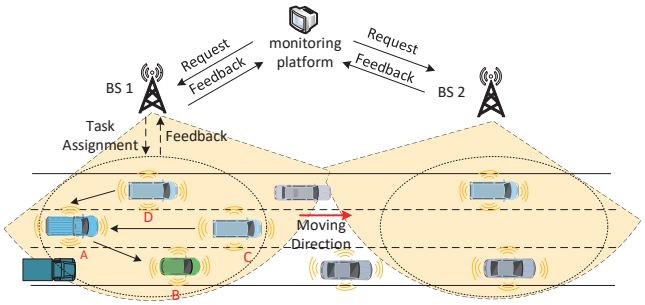


Fig. 1. The components of a monitor system for the road detection.

Fig. 1 shows the typical scenario for the highway traffic monitoring system. The entire system consists of the monitoring platform, BSes, and SVs. The BSes are connected to the monitor system, which acts as the centralized scheduler. Those SVs in the coverage area of a BS can be scheduled by the monitor system. Moreover, the SVs are able to communicate with each other with VANETs. When a detection request for the coverage area of a BS is received, the monitor system divides the detection job into several independent tasks, and assigns them to those SVs in the area. Each SV will perform data collection and computations based on the received task.

Deke Guo is the corresponding author
E-mail: {lijiangfan14, caoxiaofeng10}@nudt.edu.cn, {guodeke, xiejunjie06, chh0808}@gmail.com

Once tasks are completed, the feedback will be aggregated by one of the SVs in the field. The corresponding SV is called the **sink SV**, which will transmit the aggregated result to the monitoring platform through the BS. For example, when the monitor platform wants to know if there is an accidental obstacle on the highway under the coverage area of BS 1 in Fig. 1, the four SVs (i.e. SV A, SV B, SV C and SV D) in the area will collect the data from different sections of the highway and execute the related analysis. Last, an SV (here is SV B in Fig. 1) will be scheduled to aggregate the final result and return it to the monitoring platform through the BS 1.

Vehicular Cloud enables real-time traffic monitoring for anomaly detection and hence can help the drivers be aware of the road status ahead. However, the blind area could exist between any two neighboring BSes and will further render monitoring more difficult, due to:

- First, the distance between a pair of SVs is dynamic. When the distance from one SV to the sink SV is out of the communication range, the result output will fail to be transmitted to the sink SV.
- Second, the sink SV could move out of the coverage area of BSes before it finishes a certain task. Therefore, the SV cannot send the aggregated feedback to the monitoring platform through the related BS.
- Third, although SVs in the blind area can detect the road environment, the monitoring platform cannot request a detection job to these SVs. The communication problem in the blind area needs to be solved.

To solve those problems, we propose a novel task scheduling framework with the assistance of Unmanned Aerial Vehicles (UAVs). We use the UAV mainly for two reasons. First, UAV has already been widely used in improving cellular coverage [10], multimedia in smart cities [11] and spatial data collection [12]. They are proven to be efficient for the dynamic service demands. However, those applications are only functional for a specific area or spot, and restrained by the energy cost in cameras, communication, and computation. In this paper, UAVs only provide the feature of communication and can work in the gap with low-energy consumption. Second, compared with making a one-shot deployment of BSes, UAVs can achieve a lower cost and more flexible deployment. Moreover, due to the limited endurance of UAVs, some existing studies [13], [14] have deeply discussed the scheduling problem of UAVs, which is beyond the scope of our work.

In this paper, we utilize SVs on the highway to detect the road environment for monitoring potential anomalies. The mobility of vehicles will cause that the total available resource in the coverage area of a BS is continuously changed. Therefore, a detection job can be decomposed into several tasks, which can be processed cooperatively by multiple SVs with heterogeneous capacities and mobilities. However, there exist some blind areas without the coverage of any BS and SVs cannot communicate with the BS in these areas. Our systematical scheme, Teso, introduces UAVs to provide seamless task-offloading scheduling for those blind areas on the highway. We first formulate the task scheduling problem for the road detection under the coverage area of BSes,

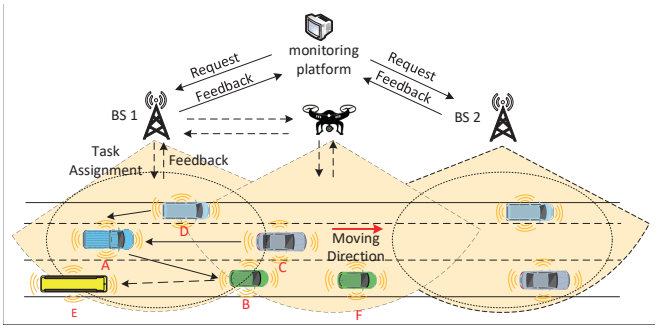


Fig. 2. An UAV-assisted monitoring system for the road detection in the highway scenario.

which is an NP-hard problem. After that, we propose an approximation algorithm to optimize the completion time of those detection requests. The key insight is to schedule tasks to SVs in different covered areas. Furthermore, we also formulate the task rescheduling scenario with the UAV assistance when the SV with a task moves into the blind area. In this case, we design a new algorithm to reschedule the task to another SV or to employ a UAV to transmit the feedback to the prior BS and the rescheduling algorithm of Teso will compare these two rescheduling solutions then make the best decision. With the extensive experiments of Teso, the scheduling algorithm can reduce the response time of detection jobs and the rescheduling algorithm can solve the blind problem within the blind area.

In summary, we make the following major contributions:

- We propose a UAV-assisted systematical scheme, named Teso, to the road detection with VCC on the highway. Teso can not only deal with the scheduling problem within the coverage area of BSes, but also solve the rescheduling problem within the blind areas.
- Teso consists of two stages. At the first stage, we design an approximation algorithm to solve the NP-hard task allocation problem of BSes with a guaranteed approximated ratio, which is NP-hard. Then at the second stage, as SVs have not finished the tasks before departing its original BS coverage area, we propose an algorithm to tackle the rescheduling problem between BSes.
- The experiment results show the efficiency and effectiveness of Teso. The scheduling and rescheduling strategy significantly shorten the average completion time of the detection requests.

The remainder of this paper is organized as follows. Section II presents system overview and problem formulation. The task scheduling solution is presented in Section III. Section IV proposes the rescheduling scheme to decide to offload the task or communicate with the UAVs. Performance evaluation is given in Section V. Section VI presents the related work of VCC and UAVs. Finally, Section VII concludes the paper and suggests our future work.

II. Task Scheduling Model and Problem Formulation

In this section, we describe the overview of the anomaly detection on the highway and the role of UAVs in the assistance.

TABLE I
MAIN NOTATIONS

$a_k(t)$	The acceleration of vehicle k at time t
acc_k	The probability for vehicle k to accelerate
AGG	The fraction of aggressive drivers
ω_r^i	The computation workload of the task r of job i
α_r^i	The input data size of the task r of job i
β_r^i	The output data size of the task r of job i
δ_r^i	The local data size of the task r of job i
dec_k	The probability for vehicle k to decelerate
D^s	The start time of the link duration
D^e	The end time of the link duration
$[-D,A]$	The range of acceleration and deceleration
$\mathscr{J}$	The set of detection jobs
Q^i	The set of tasks of job i
H^i	The dependency matrices of job i
$S_{k,j}$	The distance between vehicle k and j
R	The communication range in the VANETs
p	The fluctuation of acceleration and deceleration
P_k	The transmission power of vehicle k
P_n	The background noise power
$v_k(t)$	The velocity of vehicle k at time t
$[V_{min},V_{max}]$	The velocity range of vehicles on the highway
W	The channel bandwidth in VANETs

A. System Overview

In this paper, we consider a UAV-assisted monitoring system for the road detection in the highway scenario as illustrated in Fig. 2. Those vehicles move at high speeds on the highway, and the monitoring platform can get the information about vehicular conditions, i.e., the arrival/departure time and the computing capability of each vehicle in the VCC. Based on the information, we can model the links between vehicles [15]. Those SVs in VCC have powerful computing capability and can cooperatively complete computation services. In particular, first, the monitoring platform will receive a request to require the information about a section of the highway. Then, according to the request, the monitoring platform will generate a detection job and divide it into several tasks. Different tasks corresponding to detections of different parts of highway section. Subsequently, those tasks will be assigned to different SVs through a BS which covers the detection area. Because of the inter-independency between the tasks, some tasks need the output data of other tasks as their input data in the task executing process. After the last task of the detection job is finished, the relevant vehicle will finally upload the detection result to the monitoring platform through the BS. Table I gives the main notations in our model.

The scenario in Fig. 2 is simulated as the next moment of the scenario in Fig. 1 with the assistance of UAVs. The UAV is hovering over the gap area between BS 1 and BS 2 and can communicate with near BSes directly. In this scenario, we can see that SV B is moving into the gap area between BS 1 and BS 2, which is the blind area to the monitoring platform and BSes. It will start the computation after receiving the output data from SV A. At this time, the monitoring platform can find that SV B is moving to the gap area and cannot continue to communicate with the monitoring platform through the BS 1 right now. That is, SV B has collected the road information but has not finished the task when it has left the coverage area of BS 1. To finish the task, SV B should upload its output data as the detection result through the BS 1. However, SVs in the gap cannot communicate with the BS. There are two solutions to this problem. The first one is that SV B uploads the result to the UAV, and the UAV relays the data to BS 1, which would increase more communication latency. The second method is to offload the task that is assigned to SV B before. For example, data collected by SV B also need to be delivered to SV E under the coverage of BS 1. In this case, we say that the task is rescheduled to SV E. Thus, it is an important issue for the UAV-assisted framework to decide whether to offload the task or to communicate with the UAV. In addition, with the assistance of UAVs, the road detection in the gap area between two BSes can be conducted by SV B, SV C, or SV F.

B. Highway Mobility Model

The onboard resources of vehicles in the coverage area of the BS/UAV are not fixed due to the high mobility of SVs on the highway. For instance, the vehicle stays at the coverage area within a BS for only a few dozen seconds. Meanwhile, the capacities of different SVs are also heterogeneous. Therefore, the total available resources of vehicles in the coverage area within a BS/UAV always change dynamically over time. To fully utilize the onboard resources and schedule tasks well, it is essential to focus on the residence time of vehicles in the coverage area. Moreover, unlike other roads, the velocity of vehicles on the highway is limited in the interval of $[V_{min},V_{max}]$ to abide by traffic rules. Normally, the vehicle would not decrease to a full stop on the highway. Thus, vehicles on the highway are move at similar speeds with occasional acceleration or deceleration [16].

Based on the above analysis, we employ the mobility model in literature [15] to characterize the mobility of vehicles. The highway mobility model is a discrete time model with vehicles recalculating their acceleration every Δt seconds. For each vehicle k, its next speed is based on its current speed $v_k(t)$ and the acceleration $a_k(t)$ by the Equation (1).

$$v_k(t+\Delta t) = v_k(t) + a_k(t)\cdot\Delta t. \tag{1}$$

The acceleration of vehicle k at time t, i.e., $a_k(t)$, will be calculated by Equation (2). X_1, X_2, X_3, and X_4 are uniformly distributed random variables between 0 and 1, which model the unpredictability of the individual speed. Each vehicle has a range of acceleration, which is denoted by $[-D,A]$. A and D here are both positive and A is the maximum acceleration, while D is the maximum deceleration.

$$a_k(t) = \begin{cases} X_2\cdot A & X_1 < acc_k + p, \\ -X_1\cdot D & acc_k + p \le X_1 < acc_k + dec_k + 2p, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In the above conditions of satisfaction, $acc_k + p$ and $dec_k + p$ are the probabilities of a vehicle k to accelerate

and decelerate, respectively. The most recent researches on autonomous driving or intelligent vehicles are based on the mobility of human drivers for simulation and experiments. Comparing with the moving of SVs, the mobility with drivers can be more unpredictable with individual preferences. To improve the robustness of our task scheduling model, we refer the mobility features [17] of human driving in our paper. Particularly, acc_k and dec_k denote the probabilities for a driver to accelerate or decelerate according to personal behaviors and traffic conditions. Their values are calculated by Equation (3) and (4). All vehicles randomly accelerate or decelerate with a probability p. The higher the value of p is, vehicles are more inclined to change speeds. AGG is a parameter to denote the fraction of drivers who are aggressive (preferring either acceleration or deceleration rather than keeping a random motion around the mean velocity). Highway traffic studies suggest that about 75% of aggressive drivers tend to favor acceleration over a general mean velocity, which is used above for setting values of acc_k and dec_k [18].

$$acc_k = \begin{cases} X_4(1-2p) & X_3 < 3AGG/4, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$dec_k = \begin{cases} X_4(1-2p) & 3AGG/4 \le X_3 < AGG, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

These SVs are designed to obey the traffic rules. Those rules will constrain that the velocity of vehicles is between $[V_{min}, V_{max}]$. Thus, we have Equation (5).

$$v_k(t) = \begin{cases} V_{max} & v_k(t) > V_{max}, \\ V_{min} & v_k(t) < V_{min}. \end{cases} \tag{5}$$

R is used to denote the maximum communication distance between two vehicles in the VANETs. $S_{k,j}(t)$ denotes the distance between vehicles k and j. Thus, two vehicles can only communicate with each other when $S_{k,j}(t) \le R$. The start time $D^s_{k,j}$ and end time $D^e_{k,j}$ of the link duration between vehicle k and j can be estimated as

$$D^s_{k,j} = \arg\min_t \quad S_{k,j}(t) \le R, \tag{6}$$

and

$$D^e_{k,j} = \arg\max_t \quad S_{k,j}(t) \le R, \tag{7}$$

respectively. Specially, $D^s_{k,j} = 0$ and $D^e_{k,j} = \infty$ when $k = j$. When $D^e_{k,j} < D^s_{k,j}$, the situation indicates that no link exists between vehicles k and j.

C. Task Model

When the monitoring platform requests N detection services through those BSes as job set $\mathscr{J} = J_1, J_2, ..., J_N$. Job J_i can be divided into m_i tasks, i.e. $Q^i = q^i_1, q^i_2, ..., q^i_{m_i}$ where each task can be computed by SVs with onboard resources. In reality, not all tasks are independent mutually. When two tasks are mutually independent, they can be computed in parallel. In other case, the job is divided into tasks that need the inputs which are the outputs of other tasks. In this paper, we use a upper triangular matrices $H^i = (H^i_{l,r})_{m_i \times m_i}$ to represent the inter-dependency of tasks for job $J_i \in \mathscr{J}$, where $H^i_{l,r} = \{0,1\}, 1 \le l < r \le m_i$. $H^i_{l,r} = 1$ when task r needs the result of task l. If these m_i tasks are independent mutually, the matrices of job H^i is a zero matrices. There are three task-flow graphs shown in Fig. 3. They describe different inter-dependency relations between tasks. For instance, as shown in Fig. 3(b), task 3 can only be computed after receiving the outputs from task 1 and task 2 while task 4 needs the output from task 3. Their correspondent matrices can be given by

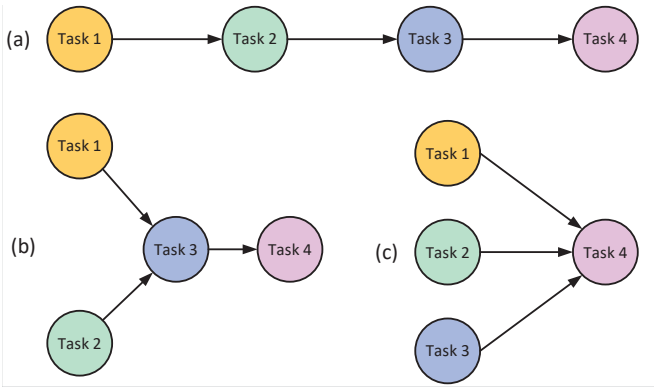


Fig. 3. Three task-flow graphs present different inter-dependencies between tasks.

$$H^1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$H^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, H^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

To describe the parametric context of each task, we define a tuple representation as $\phi^i_r = (\omega^i_r, \alpha^i_r, \beta^i_r, \delta^i_r)$. $\omega^i_r, \alpha^i_r, \beta^i_r$ and δ^i_r denote the computation workload, the input data size, the output data size and the local data size of task r of job J_i, respectively. The local data (collection data) β^i_r in a vehicle is collected by its own sensors. Thus, the total computation workload of job J_i is $\sum_{r=1}^{m_i} \omega^i_r$. As shown in Fig. 3(a), tasks of job 1 are linear logic relations. The input data α^1_{r+1} of task $r+1$ is equal to the collected data β^1_r of task r. Similarly, in Fig. 3(c), there exists $\alpha^3_4 = \beta^3_1 + \beta^3_2 + \beta^3_3$. Thus, we have

$$\alpha^i_{r+1} = \sum_{l=1}^{r} H^i_{l,r+1} \cdot \beta^i_l. \tag{8}$$

The attributes $(\omega^i_r, \alpha^i_r, \beta^i_r, \delta^i_r)$ of each task are actually determined by the assigned vehicle. Because different vehicles are distributed in different locations on the highway, the size of collection data is related to the length of the highway that the vehicle is responsible to detect. Moreover, the workload and the output data are determined by the local data. Therefore, the selection of vehicles affects the division of the job.

By employing the VANETs, we assume vehicles can communicate with its nearby vehicles and apply a link model to describe the communication between vehicles [15]. In this paper, we use the path-loss based model to measure the data transmission rate $r_{k,j}(t)$ between vehicles k and j at time t as Equation (9) shows.

$$r_{k,j}(t) = W \log(1 + \frac{P_k^t}{[S_{k,j}(t)]^h P_n}), \quad (9)$$

where W is the channel bandwidth, P_k^t is the transmission power of vehicle k, h is the path loss exponent, and P_n is the background noise power. $S_{k,j}(t)$ is the distance between vehicle k and j at time t. These two vehicles can only communicate with each other when $S_{k,j}(t) \leq R$. Recall that R is the maximum communication distance between two vehicles in the VANETs.

D. Problem Formulation

In order to calculate the completion time of a certain job, we first model the execution model of a single job. For each vehicle assigned with a task, its process consists of collecting data, computing tasks and delivering data. Data for tasks can be classified into the input data from others and data collected by its own sensors. A vehicle collects the environment data independently, which means it would not be affected by other vehicles. The collection rate of vehicle k is fixed given by λ_k and only related with the sensors. The collection time $t_{c,r}^{k,i}$ for task r of job J_i by vehicle k can be calculated based on the local data volume.

$$t_{c,r}^i = \frac{\delta_r^i}{\lambda_k}. \quad (10)$$

As for the input data, the receiving process of vehicle k is the delivering process of vehicles whose outputs vehicle k needs. Thus, we define the receiving time $t_{d,l,r}^i$ as the delivering time, which is the cost time consumed by task r to receive the output data of task l as the input data.

$$t_{d,l,r}^i = \frac{\beta_l^i}{r_{j,k}(t)}. \quad (11)$$

Here, task r and l are assigned to vehicle k and j.

We consider that different vehicles may have different computing capabilities like clock frequencies. The computing time $t_{f,r}^i$ of task r of job i processed on vehicle k is given by

$$t_{f,r}^i = \frac{\omega_r^i}{C_k}, \quad (12)$$

where C_k represents the computing capability of vehicle k.

Without loss of generality, we consider the BS/UAV schedule tasks at $t_0 = 0$. Each vehicle starts to collect environment data with their own sensors. After that, vehicles can compute tasks if they need only the local data. Otherwise, the vehicle that needs the output of other vehicles starts to wait for the data needed. We denote the starting time of computing task r of job i by T_s. If task r does not need the output from other tasks, $T_{s,r}^i = t_{c,r}^i$; otherwise, we have

$$T_{s,r}^i = \max\{H_{l,r}(T_{e,l}^i + t_{d,l,r}^i) \mid 1 \leq l \leq m_i\}, \quad (13)$$

and the ending time $T_{e,r}^i$ of task r for job i can be calculated as

$$T_{e,r}^i = T_{s,r}^i + t_{f,r}^i. \quad (14)$$

With the ending time of each task, we can calculate the time cost for finishing the job. The monitoring platform requires the status of multiple sections of the highway simultaneously. To reduce the time cost, our objective is to minimize the average job completion time by scheduling tasks to vehicles. Furthermore, the scheduling problem can be given by

$$\min \quad \frac{1}{N} \sum_{i=1}^{N} \max_{1 \leq r \leq m_i} \{T_{e,r}^i\}, \quad (15)$$

$$\text{subject to} \sum_{r=1}^{m_i} \sum_{k=1}^{K} x_{r,k}^i = 1, \forall i \in N, \quad (16)$$

$$\sum_{i=1}^{N} \sum_{r=1}^{m_i} x_{r,k}^i \cdot \omega_r^i \leq C_k, \forall k \in K, \quad (17)$$

$$D_{k,j}^s \leq t_{f,l}^i, \quad \text{if } H_{l,r}^i = 1, \forall 1 \leq r \leq m_i, \quad (18)$$

$$t_{f,l}^i + t_{d,l,r}^i \leq D_{k,j}^e, \quad \text{if } H_{l,r}^i = 1, \forall 1 \leq l, r \leq m_i, \quad (19)$$

$$x_{r,k}^i = \{0,1\}, \quad (20)$$

where $x_{r,k}^i$ denote whether task r of job i is assigned to vehicle k. Constraint (16) ensures that each task is scheduled to vehicles only once. Constraint (17) ensures that each vehicle can compute its assigned task. Constraint (18) and (19) ensure two vehicles communicate only when their distance is not more than their maximum communication distance R. The task allocation problem can be reduced to a scheduling problem as follows: a job consists of many tasks, and the task arrival time corresponds to the completion time of related tasks. These tasks can be allocated to different servers with computing capacities. According to [19], this kind of scheduling problem is NP-hard. In this case, the vehicles' mobility and communication range will further complicate this problem. Therefore, the task scheduling problem in this paper is also NP-hard, and there is no time algorithm can find the optimal solution in polynomial time.

Algorithm 1 Scheduling initialization

Input: The parametric context of each task in each job and the inter-dependency matrices of tasks $\{\Phi^i = \{\phi_r^i \mid r = 1,...,m_i\}, H^i\}_{i \in N}$, the mobility parameters of each vehicle $\{X_k, v_k, AGG_k, C_k\}_{k \in K}$

Output: The initial task placement schemes $\Theta_o = \{\theta_j^i \mid j = 1,...,m_i\}_{i \in N}$

```
i ← 1
for i ≤ N do
    while ∃(θ_1^i, ..., θ_j^i) satisfies (19) - (23) do
        Filter vehicles set K_i in the coverage of job i
        (θ_1^i, ..., θ_j^i) ← arg max_{θ_j^i ∈ K_i} Σ_{j=1}^{m_i} C_{θ_j^i}
    i ← i + 1
Return Θ_o
```

III. Efficient Scheduling Scheme

The hardness of the optimal solution motivates us to find efficient suboptimal solutions. We propose our systematical scheme, Teso, to solve the optimization problem. In this section, Teso proposes an approximation algorithm to solve the NP-hard task scheduling problem. We also prove that the

Algorithm 2 Task-vehicle scheduling optimization, Teso

Input: The parametric context of each task in each job and the inter-dependency matrices of tasks $\{\Phi^i = \{\phi_r^i \mid r = 1,...,m_i\}, H^i\}_{i\in N}$, the mobility parameters of each vehicle $\{X_k, v_k, AGG_k, C_k\}_{k\in K}$

Output: The task placement $\Theta^* = \{\theta_j^i \mid j = 1,...,m_i\}_{i\in N}$

1: Initialize $\theta^i \leftarrow \Theta_o^i$ (Algorithm 1)
2: $i \leftarrow 1$
3: **for** $i \leq N$ **do**
4: Calculate the initial response time T_i according to Θ_i
5: **for each** task dependency (q_i^a, q_i^b) in job i **do**
6: **while** $T_{r,a}^i > t_{c,b}^i$ **do**
7: Change the vehicle $k' \in K_i \setminus \theta^i$ of task q_i^a with constraint (9), (10)
8: Calculate the response time T_i'
9: **if** $T_{r,a}^i \leq t_{c,b}^i$ and $T_i' \leq T_i$ **then**
10: Update Θ_o
11: $i \leftarrow i+1$
12: Return Θ^*

approximation algorithm has a constant-factor approximation guarantee.

A. Design of Scheduling Scheme

According to the inter-dependency of tasks, some tasks need the outputs of other tasks before they are processed. If a vehicle has collected the environment data but has not received the outputs it needs, this vehicle has to wait for the input data until the completion of receiving process. For example, in Fig. 3(a), the vehicle assigned with task 2 finishes the collection process but task 1 has not been completed. In this case, the vehicle assigned with task 2 has to wait, which lengthens the completion time of the whole detection job. To reduce the response time, the best way is to make more workload processed in parallel. Therefore, the insight of our design, Teso, is to schedule those tasks on the premise that each vehicle does not need to wait for the required outputs from other related tasks. Furthermore, the key idea is to adjust the length of the highway detected by different vehicles.

Before deploying the scheduling scheme, it is essential to initialize the placement schemes for each job. The target of initialization is to find vehicles with maximum computing capacities under the constraint (19)-(23). The initialization of the task placement scheme is conducted by Algorithm 1 with a greedy strategy. For the tasks of a job, they are scheduled to vehicles through the BS. The BS can only deliver those tasks to vehicles under its coverage area (line 4 in Algorithm 1). Therefore, the initial task placement scheme will achieve the maximum computing power for task computing without considering the latency constraint.

Furthermore, Algorithm 2 is proposed to optimize the task placement based on the initial placement scheme. As mentioned before, we want to guarantee that each vehicle does not need to wait for the outputs. For example, there are two inter-dependent tasks (q_i^a, q_i^b) in job i. Task q_i^a should be completed and its output should be delivered to the vehicle with task q_i^b before this vehicle finishes the data collection (line 6 in Algorithm 2). With the mobility of vehicles, we can adjust the collected data size and the related computation workload of each task. That is, the section of the highway can be divided into different parts. The length of each part decides the length of the highway that a vehicle is responsible for detection and computation, which would also change the workload of each task. For each job, Teso can schedule tasks to different vehicles that can keep communication with each other (line 7 in Algorithm 2) on the premise that each vehicle does not need to wait for the outputs from other related tasks. Finally, the scheduling scheme can minimize the response time for those detection requests.

B. Performance Analysis

For every type of task-flow graph, the response time is exactly the brunch with the longest latency in the task flow graph. For example, the response time includes the time of tasks 1, 2, 3 and 4 in Fig. 3(a) while the response time mainly relies on the time of tasks 1, 3 and 4 (or tasks 2, 3 and 4) in Fig. 3(b). Here we assume that the collection rate and the transmission rate are the same. When those n tasks are linear logic relations, task q_{i+1} needs the output of task q_i. We set

$$T^* = t_c^1 + \sum_{i=1}^{n-1} t_d^i + \sum_{i=1}^{n} t_f^i, \tag{21}$$

where t_c^1 denotes the minimum collection time of task 1. t_d^i and t_f^i are indicate the minimum communication time and the minimum computing time available from all vehicles under the coverage of a BS. In this case, each process time in the job execution is the shortest. Meanwhile, the waiting time is also not included in T^*. Therefore, T^* can be achieved only in a perfect condition. It is worth noting that the computing capacities of different vehicles could be heterogeneous. That is, the optimal response time T^{OPT} is not less than the T^* (i.e. $T \geq T^{OPT} \geq T^*$). T is the response time of Teso and $T^{OPT} = T^*$ when the vehicles of the optimal solution have the maximum computing capacities. Our scheme, Teso, can realize that each vehicle does not need to wait for the outputs. Then, we can get

$$T \leq \max\{\frac{C_i^{OPT}}{C_i}\} T^{OPT}, \tag{22}$$

and

$$T^* \leq \max\{\frac{C_{max}}{C_i^{OPT}}\} T^{OPT}. \tag{23}$$

C_{max} is the maximum computing capacity of all vehicles under the coverage and C_i is the computing capacity of the vehicle for task i. Therefore, we can get the approximation ratio

$$T \leq \max\{\frac{C_{max}}{C_i}\} T^* \leq \max\{\frac{C_{max}}{C_i}\} T^{OPT}. \tag{24}$$

Based on the above analysis, we can see that our scheme, Teso, can achieve the constant approximation ratio. Furthermore, Teso can realize the optimal solution when those vehicles have the same capacity (i.e. C_{max}=C_i).

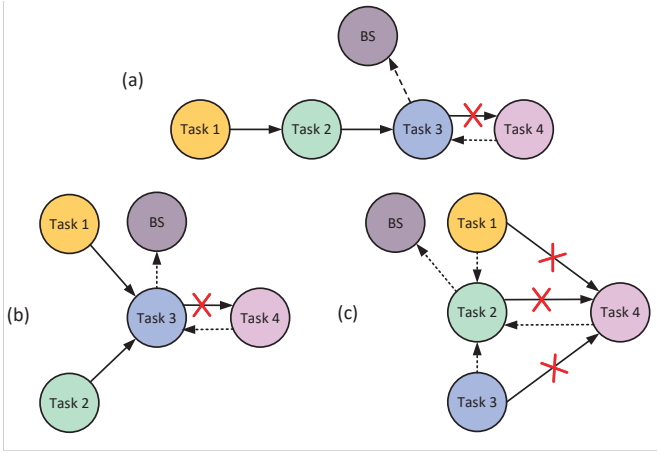


Fig. 4. The different task-flow graphs after rescheduling.

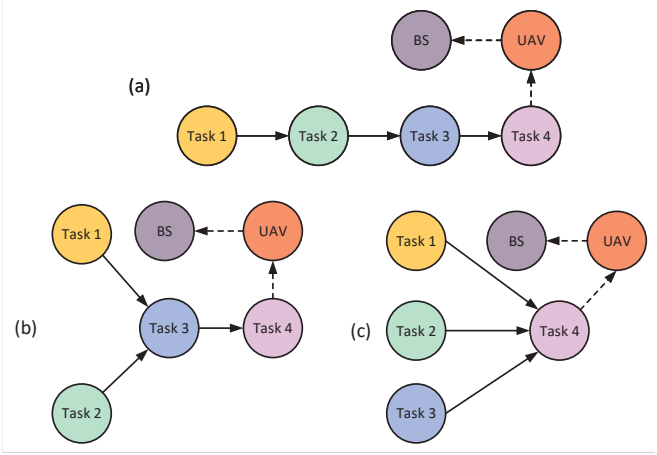


Fig. 5. The task-flow graphs by two-hop communication.

IV. Task Rescheduling

As mentioned above, the vehicles are moving fast on the highway. Thus, vehicles might leave the prior coverage area and cannot upload the task result to the corresponding BS, as it enters to new coverage areas of other BSes or blind areas. To this end, Teso can solve such the problem by rescheduling the tasks among SVs and BSes.

A. Problem Formulation

There are two approaches to upload the final result when the vehicle is out of the prior coverage area. The first one is to offload its task to another vehicle under the prior coverage area through VANETs. To be specific, the SV assigned with task r out of the prior coverage area would offload the task to vehicle j in the prior coverage area and transfer the input data α_r^i to vehicle j. This approach is shown in Fig. 4. The second one is to utilize the assistance from the UAV. The task is still processed in the vehicle before, but the result would be sent to the monitoring platform through the UAV with one more hop. More specifically, if vehicle k has left the coverage area of the BS, the main goal is to transfer the output of its task r to the monitoring system as the feedback. The second approach is shown in Fig. 5. The differences between these two approaches are the placement of processing the corresponding task r and the communication way of the final result. In the second approach, vehicle k assigned with task r keeps the computation and uploading the result as the feedback to the BS through UAVs. However, the corresponding task k in the first approach would be rescheduled to the vehicle in the prior coverage area and uploading the result to the BS directly.

For example, in Fig. 4(b), the vehicle is assigned to compute task 4 of job 2 and leaves the coverage area. Then, task 4 is reassigned to the vehicle that has been assigned to task 3 of job 2. In this case, the vehicle needs to perform computations of task 3 and task 4. Therefore, the workload of this vehicle would be heavier, and its computing time also becomes longer. Note that task 3 and task 4 are mutually dependent, and task 4 needs the output from task 3 as the input data. After the offloading of task 4, we have $\phi_3^2 = (\omega_3^2 + \omega_4^2, \beta_1^2 + \beta_2^2 + \beta_3^2 + \delta_4^2, \beta_4^2, \delta_3^2)$ and then set $\phi_4^2 = (0,0,0,0)$. As for job 3 in Fig. 4(c), the vehicle computing task 4 leaves the coverage area, and task 4 is reassigned to the vehicle originally performing task 2 of job 3. Recall that task 4 needs the outputs from task 1, 2 and 3. Thus, the local data of task 4 and the output data of task 1 and 3 should be also transmitted to task 2. In this case, we have $\phi_2^2 = (\omega_2^2 + \omega_4^2, \beta_1^2 + \beta_2^2 + \beta_3^2 + \delta_4^2, \beta_4^2, \delta_2^2)$ and set $\phi_4^2 = (0,0,0,0)$. The inter-dependency matrices change as the following matrices denoted by $\check{H}^i$.

$$\check{H}^1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\check{H}^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \check{H}^3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus, if task a is reassigned to the vehicle which is supposed to compute task b before, these two tuple representations will change into

$$\begin{cases} \omega_b^i = \omega_a^i + \omega_b^i, \\ \alpha_b^i = \sum_{l=1}^{m_i} \check{H}_{l,b}^i \cdot \beta_l^i + \delta_a^i, \\ \beta_b^i = \beta_a^i + (1 - H_{b,\cdot}^i \cdot \check{H}_{\cdot,b}^i)\beta_b^i, \\ \delta_b^i = \delta_b^i. \end{cases} \tag{25}$$

$$\phi_a^i = (0,0,0,0), \tag{26}$$

where $H_{b,\cdot}^i \cdot \check{H}_{\cdot,b}^i$ is the inner product of the b-th row of the matrices $\check{H}^i$ and the b-th column of the matrices H^i.

Hence, the optimization problem above should add two constraints of rescheduling.

$$y_{a,b}^i = \{0,1\}, \forall 1 \le a,b \le m_i, \tag{27}$$

$$H^i = \check{H}^i \text{ if } \exists\ y^{a,b} = 1, \tag{28}$$

where $y_{a,b}^i$ when task a is reassigned to task b. Constraint (28) indicates that the inter-independency matrices should be updated if there exists rescheduling.

For another approach, as shown in Fig. 5, we add the UAV and the BS as two nodes for transmitting the output from task 4 to the monitoring platform. In this way, the tuple representations $\phi_r^i = (\omega_r^i, \alpha_r^i, \beta_r^i, \delta_r^i)$. $\omega_r^i, \alpha_r^i, \beta_r^i$ do not need to

Algorithm 3 Offloading decision for vehicles out of the BS coverage area

Input: The parametric context of each task in each job and the inter-dependency matrices of tasks $\{\Phi^i = \{\phi_r^i \mid r = 1,...,m_i\}, H^i\}_{i\in N}$,the locations of vehicles $\{X_k\}_{k\in K}$, the task placement $\Theta^* = \{\theta_j^* \mid j = 1,...,\sum_{i=1}^{N} m_i\}$

Output: new placement $\breve{\Theta}$

1: $\Theta_{out} \leftarrow FindOut(\Theta^*, X_k)$
2: $num = NumOf(\Theta_{out})$
3: **for** $l \leftarrow num$ **do**
4: $\quad(\breve{H}^i, \breve{\Theta}) = OffLoad(H^i, \Theta^*, \Theta_{out}(l))$
5: $\quad T_1 \leftarrow ResponseTime(\Theta^*) + Comm(UAV, BS)$
6: $\quad T_2 \leftarrow ResponseTime(\breve{\Theta})$
7: $\quad$ **if** $T_1 > T_2$ **then**
8: $\quad\quad \breve{\Theta} \leftarrow \Theta^*$
9: $\quad l \leftarrow l + 1$

be changed. The only difference from the initial scheduling is the change of the transmission time (The final result is uploaded to the BS with one more hop). After the use of UAVs, the transmission distance from vehicle k which uploads the result to the BS has added by

$$\Delta X = \sqrt{(X_{k,0}(T_{e,k}^i) - R)^2 + (Y_{k,U})^2} - X_{k,B}(T_{e,k}^i) + X_{B,U}, \quad (29)$$

where $Y_{k,U}$ is the vertical distance between the BS and the UAV. $X_{B,U}$ is assumed fixed to denote the distance between the BS and the UAV.

Hence, the completion time of job i can be denoted by

$$T_c^i = \max\{T_{e,r}^i + \frac{\Delta X}{r_{k,B}(T_{e,r}^i)} + o \mid 1 \le r \le m_i\}, \quad (30)$$

where o denote the fixed delay time through the UAVs. Hence, the objective of the optimization problem changes into

$$\min \quad \frac{1}{N}\sum_{i=1}^{N} T_c^i, \quad (31)$$

$$\text{subject to } (16) - (20), (27), (28).$$

B. Teso for Task Rescheduling

Since some vehicles may be out of the coverage area of the prior BS before they finish their tasks, Teso further designs Algorithm 3 to decide how to reschedule those tasks. The first approach is to offload the task to other vehicles under the coverage area while the other is to upload the result through the UAV. The rescheduling algorithm compares the two approaches and chooses the better one for each task. In the pseudocode, $FindOut(Theta^*, X_k)$ is to get the placement schemes of all vehicles out of the BS coverage area according to their locations X_k (line 1 in Algorithm 3). Line 2 is to count the number of vehicles that is need to adjust. This algorithm adjusts the scheduling schemes and inter-dependency matrices after the task offloading ($OffLoad()$ at line 4 in Algorithm 3). Then, Algorithm 3 calculates the response time of task offloading ($ResponseTime()$) and the response time through the UAV ($Comm()$), respectively. Finally, Teso also decides whether to offload tasks or upload result through the UAV by comparing the response time of different schemes (lines 5-8 in Algorithm 3). It is worth noting that our rescheduling scheme can efficiently select the best rescheduling solution (offloading or UAV-assisted) to minimize the response time for those detection requests.

V. Performance Evaluation

In this section, we conduct extensive simulation-driven experiments to corroborate the efficacy of Teso.

A. Experimental Setting

First of all, we simulate a vehicular network where all vehicles move along a highway of 10 km long with 10 base stations. There are 5 gap areas between different BSes, where UAVs are hovering over these gap areas. The initial locations of vehicles are uniformly distributed along the highway, and the initial moving speed follows a uniform distribution between V_{min} and V_{max}. After that, the moving speed and locations of vehicles change according to the highway mobility model [15]. Meanwhile, the monitoring platform sends requests to detect multiple sections of the highway. The default values of the parameters used in our analysis are listed in Table II that is referred to literatures [20] and [21].

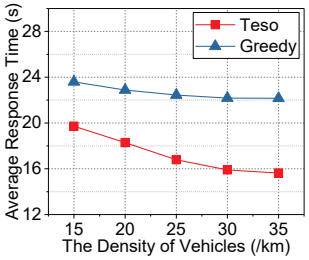
Then, we conduct experiments to evaluate the performance of our proposed Teso in VCC. We compare our Teso scheduling solutions with the original Greedy Algorithm, which focuses on the minimization of the processing time of each task. Without loss of generality, we assume the communication environments (the transmission power and the background noise power) are the same in both solutions. Furthermore, we also consider the heterogeneity of vehicular computing and collecting capabilities. The average response time of multiple experiments is calculated as the evaluation index.

B. Average Response Time

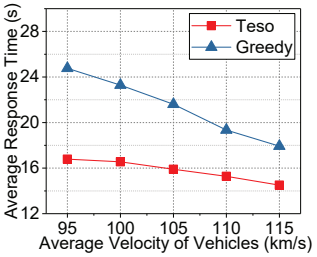
Fig. 6 shows the average response time of our proposed algorithm Teso and the Greedy algorithm. In different experi-

TABLE II
The default values of main parameters

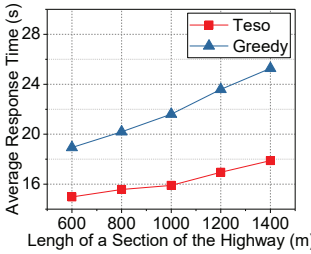
Parameters	Values
Length of a section of the highway (m)	1000
Communication range of a vehicle (m)	300
Transmission power of requestor (mW)	200
Wireless channel bandwidth (MHz)	5
Background noise power (mW)	10^{-9}
Path loss exponent	4
The range of vehicles' velocity (km/s)	[90,120]
Computing capabilities of vehicles	$c, 2c, 3c$ *or* $4c$
The number of tasks in an job	4
Computation workload of each task	$\omega, 2\omega$ or 3ω
Density of vehicles in the highway (/km)	25



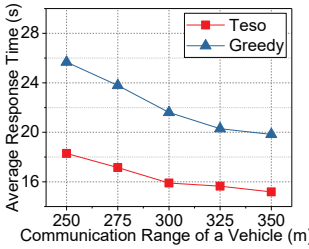
(a) The average response time under different vehicle densities.



(b) The average response time under different vehicles' velocities.



(c) The average response time under different length of a section of highway.



(d) The average response time under different communication range of a vehicle.

Fig. 6. The average response time of Teso and Greedy.

ments, we vary some parameters while keeping other parameters fixed and evaluate its variation trend. The parameters (e.g., density range, driving speed) in the experiments is based on the vehicles' safe distance on the highway and accords with the general highway conditions. Fig. 6(a) shows that with the increase of the density of vehicles, the average response time of the assigned jobs is decreasing for both algorithms. The main reason is the number of available vehicles for scheduling increases and the total resources in the VCC are increasing. Under this situation, a job has more options to be divided into different tasks. However, the Greedy has a little improvement because it only considers minimizing the processing time. Meanwhile, Fig. 6(b) shows the trend of the average response time when the average velocity of vehicles in the VCC is increased. Unlike the increase of the density of vehicles, Teso does not make such a great improvement as Greedy, but Teso still achieves better response time than the original Greedy algorithm. In the total detection time on the highway, the largest proportion of the total time is the collection time, which depends on the driving time of the section of the highway, which is longer than the processing time and the communication time.

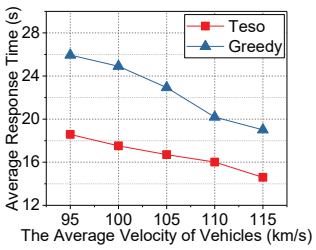
Besides, we calculate the average response time by varying parameters related to distance. In the default setting, a job detects the highway with 1 *km* sensing range. In Fig. 6(c), we vary this length from 0.6 *km* to 1.4 *km*. The average response time of Teso and Greedy is increased by about 2.9 *s* and 6.3 *s*, respectively. When a job is required to detect a longer length on the highway, the total collection distance and data are increased, which also causes heavier computing workload. Therefore, the response time will also be longer. Meanwhile, Teso performs better in optimizing the average response time and keeps the response time at a lower level. Then, we reset the length of jobs' range to 1 *km* and change the value of the communication range of a vehicle from 250 *m* to 350 *m*. As shown in Fig. 6(d), we find the decreasing trend is similar to the increasing trend in Fig. 6(c). Both changing ranges of the average response time are almost the same in the two figures. Shorter communication range of a vehicle makes less number of vehicles communicating with each other. Therefore, this causes less available schemes to schedule detection tasks, which makes the average response time longer than that in a longer communication range.

C. Distribution Types of Vehicles' Velocities

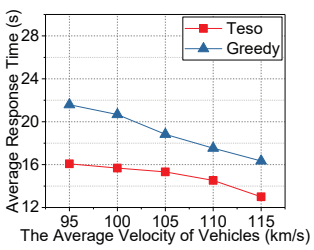
In the initial setup, the initial velocity of vehicles follows a uniform distribution between V_{min} and V_{max}. In this section, we conduct experiments of the normal distribution and the (negative) skewed distribution. Fig. 7(a) and Fig. 7(b) show the average response time when vehicles' velocities follow the normal distribution and the negative skewed distribution, respectively. From the two figures, we can see that the proposed Teso is superior to the Greedy Algorithm in terms of the response time. With the increase of the average velocity of vehicles, the average response time is decreased, as there are more vehicles moving fast and decrease the collection time. Compared to the results under the uniform distribution in Fig. 6(b), the average response time under the normal distribution is a little longer than the uniform distribution while the average response time of the skewed distribution is shorter. The results are in line with the expectations of changes in the collection time because the velocity mainly influences the collection time of the task.

D. Time division

Fig. 8 shows the collection, communication and processing time of tasks in a job, whose task-flow tasks have been shown in Fig. 3(b). The communication time is significantly less than the collection time and the processing time. The collection time is the most because it depends on the length of highway collected by the vehicle and the vehicles' velocity. It is noted that the collection time in Teso and Greedy is almost the same, which indicates these two algorithms have the similar effects on the collection operations. This is because there are



(a) The average response time with the normal distribution of vehicles' velocities.



(b) The average response time with the skewed distribution of vehicles' velocities.

Fig. 7. The average response time under different velocity distribution.

Fig. 8. Collecting , communicating and processing time of a job.

small differences in vehicles' velocity and the total length of highway is fixed. Moreover, the durations of task 1 and task 2 in Teso and Greedy have no obvious difference. Because the Greedy Algorithm tends to minimize the duration of each task and selects vehicles with maximum computation capacity, the scheduling scheme for the first two tasks is as good as the Teso. However, Greedy has the intrinsic defect to optimize the durations of subsequent tasks, which cause its scheduling scheme for tasks 3 and 4 is worse than that of Teso. Since the communication time includes the queuing time for the output of the related task, Greedy may cause tasks 3 and 4 a longer waiting time than Teso. In our approach, Teso reduces the workload of tasks 1 and 2 with less length of the highway to collect data and can reduce the queuing time in the system. From the execution process, we can see that the proposed Teso is superior to the Greedy solution.

E. Rescheduling Results

As mentioned before, we have added UAVs on the gap between two BSes for assisted communication because of the limited coverage of the BS. Some vehicles may be out of the coverage area of the BS before they finish their tasks. In order to verify the validity of the rescheduling algorithm (deciding to offload tasks or to transfer through the UAV), we compare our rescheduling algorithm with two strategies. One is always offloading tasks out of the coverage area to vehicles in the coverage area, and the other is always sending the result of tasks to the BS through the UAV. Fig. 9 shows the results of three different approaches. The average response time of our rescheduling algorithm is better than the other two strategies. Moreover, we find that with the increase of the density of

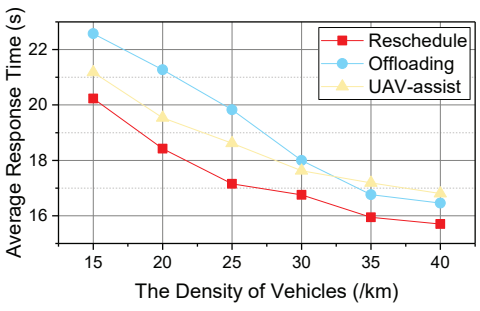


Fig. 9. The average response time after rescheduling.

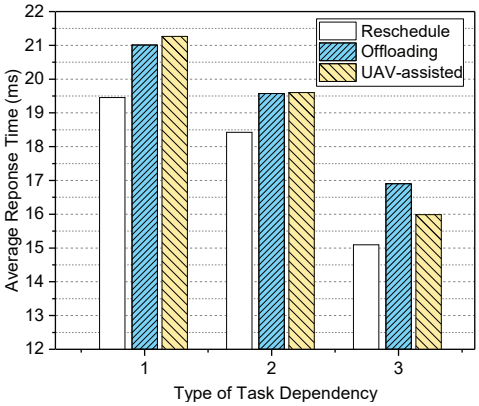


Fig. 10. The average response time under different types of task dependency.

vehicles on the highway, the offloading strategy is worse than the UAV-assist strategy at the beginning but is better when the density of vehicles reaches $35/km$. This is because there are more offloading schemes with more vehicles on the highway. This indicates that it is necessary to make a decision to offload tasks or upload results by UAVs.

F. Task Type Analysis

Fig. 10 shows the average response time when we change the types of task dependency, which have been shown in Fig. 3. We find the two strategies almost have the same results for the second type of task dependency. Besides, UAV-assist strategy is overall better than the offloading strategy for the third type of task dependency, because task 1, task 2 and task 3 are processed in parallel. Offloading strategy is better than UAV-assisted strategy for the first type of task dependency. Therefore, UAV-assisted strategy would be better if there are more parallel tasks in a job. It is worth noting that our rescheduling algorithm always has shorter response time than the two strategies because our algorithm can choose to offload the task to other SVs or communicate through UAVs with one more hop. It can be seen that the proposed rescheduling algorithm has improvements on the response time and increases the system stability.

VI. Related Work

Related work can be divided into two categories. 1) Task scheduling in VCC and 2) UAV assisted scheduling.

In the first category, the task scheduling in VCC mainly includes the applications of static scenarios [22] and dynamic scenarios [23]. As for the static scenario, vehicles with computing capacities are parking in the garages of malls, airports and uptown. These vehicles can be shared/rented as plentiful and stable resources for nearby requests, which has the similar computing functions with edge computing [25], [26]. Arif et al. studied arrival and departure of vehicles at the airport and proposed a time-dependent parking lot occupancy model to schedule computational tasks [27]. Considering the diversity of VCC resources, a semi-Markov decision processes model for VCC resource allocation is proposed for the heterogeneous resources of vehicles and roadside units. Then, a Markov decision approach is proposed to find the optimal

strategy of resource allocation for tasks [4]. In the dynamic scenario, the mobility of vehicles [28] makes it difficult for scheduling tasks to servers on SVs. Kyutoku et al. propose a method for detecting general obstacles on a road by subtracting present and past in-vehicle camera images [29], while the time-efficient and fast-responsive requirements are not considered. Literature [21] proposed a vehicle-based cloudlet relaying scheme for mobile computation offloading and minimized the energy consumption of the mobile device while satisfying the delay constraint of the application. The blind areas without the coverage of BSes or RSUs are rarely considered in previous works, however, which could significantly influence the feedback uploading from vehicles. Unlike prior work, our paper first proposes the road detection by utilizing sensors of SVs and the computing environment of VCC cooperatively. We introduce UAVs to achieve the interconnectivity between SVs and BSes within blind areas.

In the second category, the use of UAVs such as quadcopters and unmanned gliders [30] can assist in task processing and data processing. Literature [31] deployed unmanned aerial base stations (UABSes) to deal with the communication difficulties caused by malevolent attacks or natural disasters [32]. Oubbati et al. [33] made UAVs cooperate with VANET on the ground so as to assist in the routing process and improved the reliability of the data delivery by bridging the communication gap. Some existing studies already focused on the hovering of UAVs to function under a certain area. Literature [34] employed UAVs to assist VANETs and kept UAVs cover a certain area to maximize the throughput. Zhang et al. studied a wireless communication system with a fixed-wing UAV employed to collect information and aimed to maximize the UAV's energy efficiency [14]. In this paper, we focus on the communication between UAVs and VANETs of SVs within the blind areas. Prior work has studied the hovering and scheduling of UAVs, which fully shows the feasibility to deploy UAVs for communication in an area. These UAVs can be functioned with low-energy consumption and flexible locations, which can be more manageable comparing with one-shot BSes deployment.

VII. Conclusion and Future Work

In this paper, we have proposed a systematical framework, named Teso, in the scenario of real-time anomalies detection on the highway. Teso consists of two stages. At the first stage, Teso has modeled the problem of scheduling tasks in a detection job to multiple SVs on the highway. An approximation algorithm has been proposed to solve the task scheduling problem to minimize the response time. At the second stage, UAV is utilized to address the problem of blind areas without the coverage of BSes. Correspondingly, Teso has proposed a rescheduling scheme to deal with the case, where the SV moves out the previous coverage area and into the blind area. Extensive experiments have been conducted to show the efficiency of Teso. Teso is an effective systematical scheme and can be deployed in the monitoring platform conveniently, which is readily to be applied to the intelligent transportation system. In the future work, we will jointly consider the trajectory design of UAVs in the VCC tasks scheduling scenario.

Acknowledgment

This work was partially in part supported by the National Natural Science Foundation of China under Grant U19B2024 and 61772544, National key research and development program under Grant 2018YFB1800203 and 2018YFE0207600, Tianjin Science and Technology Foundation under Grant 18ZXJMTG00290.

References

[1] Q. Cui, Y. Wang, K. Chen, W. Ni, I. Lin, X. Tao, and P. Zhang. Big data analytics and network calculus enabling intelligent management of autonomous vehicles in a smart city. *IEEE Internet Things J.*, 6(2):2021–2034, Sep. 2018.

[2] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis. A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet Things J.*, 5(2):829–846, Mar. 2018.

[3] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Trans. Veh. Technol.*, 65(6):3860–3873, 2016.

[4] C. Lin, D. Deng, and C. Yao. Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units. *IEEE Internet Things J.*, 5(5):3692–3700, Apr. 2017.

[5] M. LiWang, S. Hosseinalipour, Z. Gao, Y. Tang, L. Huang, and H. Dai. Allocation of computation-intensive graph jobs over vehicular clouds in iov. *IEEE Internet Things J.*, 2019.

[6] N. Akhtar, O. Ozkasap, and S. C. Ergen. Vanet topology characteristics under realistic mobility and channel models. In *Proc. of IEEE WCNC*, pages 1774–1779, Apr. 2013.

[7] X. Cao, Y. Li, J. Han, P. Yang, F. Lyu, D. Guo, and X. Shen. Online worker selection towards high quality map collection for autonomous driving. In *Proc. of IEEE GLOBECOM*, 2019.

[8] D. T. Kanapram, F. Patrone, P. Marin-Plaza, M. Marchese, E. Bodanese, L. Marcenaro, D. M. Gomez, and C. Regazzoni. Collective awareness for abnormality detection in connected autonomous vehicles. *IEEE Internet Things J.*, 2020.

[9] O. Akoz and M. Karsligil. Traffic event classification at intersections based on the severity of abnormality. *Mach. Vision Appl.*, 25(3), Apr.

[10] U. Urosevic, Z. Veljovic, and M. Pejanovic-Djurisic. Improving cellular coverage through uavs. In *Proc. of WPMC*, pages 68–73, Dec. 2017.

[11] G. K. Garge and C. Balakrishna. Unmanned aerial vehicles (uavs) as on-demand qos enabler for multimedia applications in smart cities. In *Proc. of 3ICT*, pages 1–7, Nov. 2018.

[12] T. Yu, X. Wang, and A. Shami. Uav-enabled spatial data sampling in large-scale iot systems using denoising autoencoder neural network. *IEEE Internet Things J.*, 6(2):1856–1865, Oct. 2018.

[13] J. Zhang, L. Zhou, Q. Tang, E. C. Ngai, X. Hu, H. Zhao, and J. Wei. Stochastic computation offloading and trajectory scheduling for uav-assisted mobile edge computing. *IEEE Internet Things J.*, 6(2):3688–3699, Dec. 2018.

[14] J. Zhang, Y. Zeng, and R. Zhang. Receding horizon optimization for energy-efficient uav communication. *IEEE Wireless Commun. Lett.*, PP:1–1, Dec. 2019.

[15] Z. Wang, Z. Zhong, D. Zhao, and M. Ni. Vehicle-based cloudlet relaying for mobile computation offloading. *IEEE Trans. Veh. Technol.*, PP(99):1–1, Sep. 2018.

[16] V. Namboodiri and L. Gao. Prediction-based routing for vehicular ad hoc networks. *IEEE Trans. Veh. Technol.*, 56(4):2332–2345, Aug. 2007.

[17] Y. Zhang, H. Chen, S. L. Waslander, T. Yang, and S. Zhang. Toward a more complete, flexible, and safer speed planning for autonomous driving via convex optimization. *IEEE Sensors J.*, 18(7):2185, May 2018.

[18] V. Muchuruza and R. N. Mussa. Speeds on rural interstate highways relative to posting the 40 mph minimum speed limit. *JT&S*, 7:71–84, Jan. 2005.

[19] Z. Lu, J. Zhao, Y. Wu, and G. Cao. Task allocation for mobile cloud computing in heterogeneous wireless networks. *Proc. of ICCCN*, pages 1–9, 2015.

[20] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen. Cooperative task scheduling for computation offloading in vehicular cloud. *IEEE Trans. Veh. Technol.*, 67(11):11049–11061, Aug. 2018.

[21] Z. Wang, Z. Zhong, D. Zhao, and M. Ni. Vehicle-based cloudlet relaying for mobile computation offloading. *IEEE Trans. Veh. Technol.*, PP:1–1, Sep. 2018.

[22] F. Ahmad, M. Kazim, A. Adnane, and A. Awad. Vehicular cloud networks: Architecture, applications and security issues. In *Proc. of UCC*, Dec. 2016.

[23] T. Mekki, I. Jabri, A. Rachedi, and M. Jemaa. Vehicular cloud networks: Challenges, architectures, and future directions. *Veh. Commun.*, pages 268–268, Jan. 2017.

[24] K. Z. Ghafoor, K. A. Bakar, M. A. Mohammed, and J. Lloret. Vehicular cloud computing: Trends and challenges. *ZANCO Journal of Pure and Applied Sciences*, 28(3):67–77, Nov. 2016.

[25] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang. Edge federation: Towards an integrated service provisioning model. *IEEE/ACM Trans. Netw.* to appear.

[26] J. Xie, C. Qian, D. Guo, M. Wang, S. Shi, and H. Chen. Efficient indexing mechanism for unstructured data sharing systems in edge computing. In *Proc. of INFOCOM*, Apr. 2019.

[27] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yan, and I. Khalil. Datacenter at the airport: Reasoning about time-dependent parking lot occupancy. *IEEE Trans. Parallel Distrib. Syst.*, 23:2067–2080, Nov. 2012.

[28] K. A. Ali, O. Baala, and A. Caminada. On the spatiotemporal traffic variation in vehicle mobility modeling. *IEEE Trans. Veh. Technol.*, 64(2):652–667, Feb. 2015.

[29] H. Kyutoku, D. Deguchi, T. Takahashi, Y. Mekada, I. Ide, and H. Murase. On-road obstacle detection by comparing present and past in-vehicle camera images. In *Proc. of MVA*, Mar. 2013.

[30] N. B. Johnson. Are drones and robots the future of public safety? http://www.statetechmagazine.com/article/2014/06/aredrones-and-robots-future-public-safety. June, 2014.

[31] A. Merwaday and I. Guvenc. Uav assisted heterogeneous networks for public safety communications. In *Proc. of WCNCW*, pages 329–334, Jun. 2015.

[32] M. Erdelj and E. Natalizio. Uav-assisted disaster management: Applications and open issues. In *Pro. of ICNC*, pages 1–5, Feb. 2016.

[33] O. S. Oubbati, A. Lakas, F. Zhou, M. Gnes, N. Lagraa, and M. B. Yagoubi. Intelligent uav-assisted routing protocol for urban vanets. *Comput. Commun.*, 107:93–111, Apr. 2017.

[34] X. Fan, C. Huang, X. Chen, S. Wen, and B. Fu. Delay-constrained throughput maximization in uav-assisted vanets. In *WASA*, pages 115–126, Jun. 2018.