Satellite-UAV-MEC Collaborative Architecture for Task Offloading in Vehicular Networks

Yu-Hsiang Chao*, Chi-Hsun Chung*, Chih-Ho Hsu†, Yao Chiang†, Hung-Yu Wei† and Chun-Ting Chou*
* Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan
† Graduate Institute of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
{hywei, chuntingchou}@ntu.edu.tw

***Abstract*—Mobile Edge Computing (MEC) has been proposed as a solution for relieving the network congestion problem in the backhaul by distributing computing resources near to end devices, providing a chance to satisfy low latency requirement applications. Unmanned aerial vehicle (UAV), on the other hand, has been investigated to be a flying edge node. With its high mobility, it can adapt well to any kind of scenarios such as highway, city, or rural. In this paper, we propose an architecture composed of a satellite, MEC servers connected to base stations (BSs), UAVs, together with vehicular networks. Under this scheme, we propose a two-stage algorithm called SUM by considering the UAVs deployment and task offloading decision, aiming at accepting more requests and enhance the resource utilization. The simulation results show that our proposed algorithm outperforms the other three approaches in terms of long-term profit.**

***Index Terms*—Mobile edge computing (MEC), Task offloading, Unmanned aerial vehicle (UAV), Vehicular network.**

I. Introduction

With the rapid development of mobile technology and the ever-increasing number of vehicles, more and more novel vehicular applications emerge, such as assisted and autonomous driving, speech recognition, or vehicle to vehicle communications, and all of them require a large amount of computing resource for the data processing with low latency [1]–[3]. To deal with the diverse application demands of the vehicles, a cloud-based mobile edge computing (MEC) offloading framework in vehicular networks was proposed [4]. Compared with the traditional cloud-enabled vehicular network which may incur high latency not only due to the long distance between vehicles and the cloud servers, but also the backhaul network congestion problem when there's a high demand for applications, MEC offloading scheme solves these problems by providing computing resources close to the terminals [5].

Even though the low latency requirements of the vehicular applications can be satisfied with the MEC servers, the resource limitation in each edge node causes another problem. Specifically, since resources in edge are often deployed in a distributed manner, and are relatively limited compared with those in the cloud environment, how to orchestrate and allocate resources properly to enhance the utilization becomes a critical issue in MEC networks [6]–[8].

Aside from the MEC servers located near base stations (BSs), unmanned aerial vehicles (UAVs) can also act as flying edge nodes, providing computation resource near to the vehicular terminals [9]–[11]. In [12], the authors jointly optimized the trajectory of UAV and the allocation of computing resources to minimize the largest energy consumption among mobile terminals. In [13], the authors proposed a multi-UAV-enabled MEC system for mobile users with a two-layer optimization method called ToDeTaS, aiming at minimizing system energy consumption by jointly considering the deployment UAVs resource allocation. Compared with the MEC servers fixed to the BSs which limit the capability of MEC, UAVs can be controlled remotely and have higher mobility to be deployed flexibly and dynamically, and thus often be used to extend the coverage of a wireless system with some applications such as military scenarios, rescue and emergency search, disaster response [14]–[16]. Even though there are many studies investigated the tasking offloading problem for vehicular networks [17], [18] as well as UAV-enabled MEC systems for mobile users or IoT devices [12], [13], [19], [20], few works focus on the UAV enabled MEC system together with the vehicular networks, where the vehicular tasks can be executed locally in the vehicles, or be offloaded to a UAV or an MEC server.

In this work, we try to increase the long-term profit by introducing UAVs as other resource providers, integrated with MEC servers at the BSs. Specifically, we consider a satellite to be an orchestrator, for allocating resources effectively and efficiently with a global view. Our contributions of this work can be summarized as follows:

1) We propose an integrated architecture consisting of a satellite, MEC servers, and UAVs for vehicular networks, where the satellite has a global view for the whole network and acts as an orchestrator for resource coordination among the vehicles. We believe that this architecture can be deployed in the future network.

2) A two-stage optimization approach called SUM is proposed, with the purpose of maximizing the long-term profit by optimizing the distribution of UAVs and the computational resources allocation. The positions of UAVs are dynamically adjusted to the densely populated areas in the first stage which increases the chance of accepting more higher profit tasks. The near-optimal task assignment is then performed in the second stage, aiming at maximizing the profit for each timeslot.

The rest of the article is organized as follows. In Section II, we provide the proposed system model. In Section III, the problem formulation is given. The proposed algorithm and the simulation results are presented in Section IV and

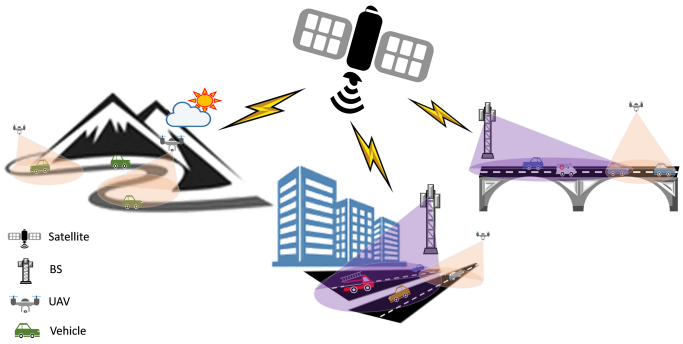

Fig. 1: System model of SUM

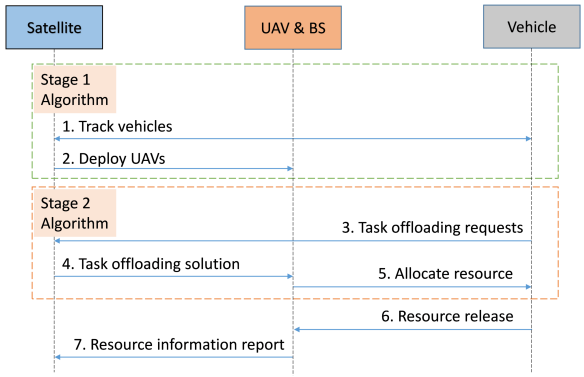

Fig. 2: Message workflow of SUM

V, respectively. Finally, we conclude this paper in Section VI.

II. System Model

As shown in Fig. 1, we consider an MEC system consisting of v vehicles denoted as $V = \{1, 2, \ldots, v\}$, n UAVs denoted as $N = \{1, 2, \ldots, n\}$, and m base stations (BSs) denoted as $M = \{1, 2, \ldots, m\}$, and a low earth orbit (LEO) satellite. The LEO satellite has lower propagation delay, and can act as an orchestrator. By collecting the information for all the vehicles, it can efficiently assign UAVs to high vehicular density regions. Once receiving offloading requests from vehicles, the satellite will decide whether to allocate resource to them or not according to its algorithm. Since it has a global view for the whole network, the resource can be utilized more efficiently, and further increases the revenue. The overall workflow is presented in Fig. 2.

For each BS $i \in M$ is equipped with an MEC server with N_{MEC} CPU cores through wired connections, and the MEC server collects tasks from its attaching BS directly. Each MEC server core has computing frequency f_{MEC}. The coordinate of the BS i is $\{X_i, Y_i\}$, and R_{BS} is the radius of service coverage of the BS.

For each UAV $j \in N$, we assume that it also has N_{UAV} CPU cores and each core has the computing frequency f_{UAV} for serving some vehicles in its service area. The horizontal radius of service coverage of the UAV is R_{UAV}. The locations of these UAVs are controlled by the satellite, and are denoted as $\{X_j(t), Y_j(t)\}$, where t represents the t-th timeslot. Since the satellite has the information about each vehicle, it can assign UAVs to the region where might need more computing resources. When the UAV is changing its position, it cannot serve any tasks. Here we assume that the flying velocity is large enough so that the flying time can be approximate to 0. If there are still some tasks unfinished but the UAV j changes its position, it will incur a punishment due to the failure of the task.

For each vehicle $s \in V$, we use a tuple $(x_s(t), y_s(t), v_s(t), \theta_s(t), q_s)$ to represent its status at timeslot t. The first two parameters represent the position of the vehicle at time t. $v_s(t)$, $\theta_s(t)$ represent speed and direction respectively. In addition, we assign different priority level q_s to each vehicle. Emergency vehicles such as ambulances, police cars, or fire trucks, are labeled as high-level vehicles. Since most of their tasks are urgent, we will try to ensure their tasks to be executed. All of the vehicles have the same computing capacity f_V, and can only execute their own tasks.

The task generated by a vehicle $s \in V$ at timeslot t is denoted by $U_s(t) = (a_s, c_s, T_s^{max})$, where a_s denotes the input data size of the task. c_s is the computational resource requirement for completing the task, T_s^{max} is the delay tolerance of the task.

In our system, each task can be executed locally, offloaded to a UAV or an MEC server. Note that we consider full offloading here. For the task $U_s(t)$, we denote the actual delay as T_{U_s}. In the following subsections, we will introduce two execution schemes which are local execution and remote execution.

(i) Local Execution: We denote the local task completion time $T_{U_s}^l$ as

$$T_{U_s}^l = \frac{c_s}{f_V} \tag{1}$$

If the task $U_s(t)$ from the vehicle $s \in V$ is executed locally, it must satisfy:

$$T_{U_s}^l \leq T_s^{max} \tag{2}$$

(ii) Remote Execution: There are two schemes for remote execution, which are UAV execution and MEC execution. For simplicity, we refer to both UAVs and MEC servers as remote servers.

In this scheme, the remote task completion time $T_{U_s}^r$ will be divided into two parts: execution time $T_{U_s}^c$ and data transmission time$T_{U_s}^t$. Thus,

$$T_{U_s}^r = T_{U_s}^c + T_{U_s}^t \tag{3}$$

$$T_{U_s}^c = \frac{c_s}{f_j} \tag{4}$$

$$T_{U_s}^t = \frac{a_s}{r^{v,j}} \tag{5}$$

where $r^{v,j}$ is the data transmission rate between the vehicle and the j-th remote server, and is given by:

$$r^{v,j} = W \cdot log_2(1 + SINR) \tag{6}$$

TABLE I: NOTATION LIST

Notation	Definition
V	A set of vehicles
N	A set of UAVs
M	A set of BSs
N_{MEC}	Number of CPU cores in each MEC server
N_{UAV}	Number of CPU cores in each UAV
f_{MEC}	Computing frequency of the MEC core
f_{UAV}	Computing frequency of the UAV core
f_V	Computing frequency of the vehicle
$x_s(t), y_s(t)$	Coordinate of the vehicle s at time t
$v_s(t)$	Speed of the vehicle s at time t
$\theta_s(t)$	Direction of the vehicle s at time t
q_s	Priority level of the vehicle s
$U_s(t)$	The task generated by the vehicle s at time t
a_s	Input data size of the task generated by vehicle s
c_s	Computational resource requirement of the task generated by vehicle s
T_s^{max}	Delay tolerance of the task generated by vehicle s
T_{Us}	Actual delay of the task U_s
$T_{U_s}^{\ l}$	Local task completion time
$T_{U_s}^{\ r}$	Remote task completion time
$T_{U_s}^{\ C}$	Execution time for remote execution
$T_{U_s}^{\ t}$	Transmission time for remote execution
$r^{v,u}$	Data rate between the vehicle and the UAV
$r^{v,b}$	Data rate between the vehicle and the BS
$d_{s,j}$	Decision variable for the s-th vehicle
P_{MEC}	Power for executing a task on an MEC server
P_{UAV}	Power for executing a task on a UAV

where W is the size of sub-bands for each vehicle. We assume that the UAV and BS adopt different frequency bands, so there is no interference between UAVs and BSs. In addition, we consider OFDMA in the uplink scheme, so the interference between vehicles can be avoided too.

If the task $U_s(t)$ from the vehicle $s \in V$ is offloaded to the j-th remote server, it must satisfy:

$$T_{U_s}^{\ r} \leq T_s^{max} \tag{7}$$

$$T_{U_s}^{\ r} < T_{U_s}^{\ l} \tag{8}$$

$$[x_s(t)-X_j(t)]^2+[y_s(t)-Y_j(t)]^2 \leq R_j^2, \forall t \in [t_{arr}, t_{arr}+T_s^{max}] \tag{9}$$

where t_{arr} is the arriving time of the task.

We assume that all of the offloading queries are rational. That is, no one is willing to pay for an inferior service with high latency. Therefore, we add a constraint to ensure that all of the offloaded tasks will spend less time than executed locally. In addition, for the tasks being offloaded, we must ensure that the vehicles stay in the coverage of the UAV or the BS for receiving the results back. These two constraints are described in equation (8) and (9) respectively.

We define a decision variable $d_{s,j} \in \{0,1\}$, where the subscription s represents the s-th vehicle, and j represents the j-th server. That is, if the task is executed by itself, $d_{s,0} = 1$, and $d_{s,j} = 0$ for $j \neq 0$. Similarly, $d_{s,j} = 1$ for $j \in \{1, \ldots, n\}$ represents the task s is offloaded to the j-th UAV, and $d_{s,j} = 1$ for $j \in \{n+1, \ldots, n+m\}$ represents the task s is offloaded to the $(j-n)$-th MEC server. Since each task can only be executed in one place, we have $\sum_{j=0}^{n+m} d_{s,j} \leq 1$. TABLE I lists out the important notations used in this work.

III. Problem Formulation

The main objective of our work is to maximize the long-term profit. Based on our previously discussed model, we define a revenue function and several cost functions including potential cost, energy cost, and contract breaking punishment, namely penalty.

A. Revenue model

We define the revenue function for completing a task as follows:

$$R_{Us}(t) = \begin{cases} q_s \sum_{j=1}^{n+m} \gamma_j \cdot d_{s,j} \cdot c_s, \forall s \in V, & \text{if } T_{Us} \leq T_s^{max} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where q_s is the priority level of the vehicle s. γ_j is the charge per CPU capacity for the j-th server. Specifically, we have two different charges γ_{MEC}, γ_{UAV} for offloading tasks to the MEC server and UAV respectively. $d_{s,j} \in \{0,1\}$ is the decision variable of a vehicle $s \in V$, and c_s is the computational resource requirement for completing the task.

Finally, the revenue of UAVs and MECs for completing tasks over a period of time T, denoted as $R(T)$, is calculated as

$$R(T) = \sum_{t=0}^{T} \sum_{s \in V} R_{U_s(t)} \tag{11}$$

B. Cost model

Here we provide several cost functions including potential cost, energy cost, and penalty. The overall cost will be the linear combination of these three costs.

1) Potential cost: For a task which can not be executed either locally or be offloaded, there will be a potential cost η which is a constant for unsatisfying a task. The potential cost for each task is defined as follow:

$$PC_{U_S}(t) = \begin{cases} \eta, & \text{if the task } U_S(t) \text{ is unsatisfied} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Then, the long-term potential cost over a period time T will be:

$$PC(T) = \sum_{t=0}^{T} \sum_{s \in V} PC_{U\,s(t)} \tag{13}$$

2) Energy cost: We define an energy cost model for executing a task as

$$E_{U_S}(t) = \begin{cases} \frac{\sum_{j=1}^{n+m} P_j \cdot d_{s,j} \cdot c_s}{F_j} + tr, \forall s \in V, & \text{if } T_{Us} \leq T_s^{max} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where P_j is the power of the j-th server per CPU core. Notice that $P_j = P_{UAV}$ for $j = 1, ..., n$; $P_j = P_{MEC}$ for

$j = n+1, \ldots, n+m$. q_s is a the priority level of the vehicle s. $d_{s,j} \in \{0,1\}$ is the offloading decision variable of a vehicle $s \in V$, and c_s is the computational resource requirement for completing the task. F_j is the CPU frequency of the j-th server. tr is the transmission energy for sending the result back to the vehicle.

The overall energy consumption for executing the offloaded tasks on UAVs and MECs over a period time T, denoted as $E(T)$, is calculated as

$$E(T) = \sum_{t=0}^{T} (P_{UAV} \cdot N'_{UAV}(t) + P_{MEC} \cdot N'_{MEC}(t) + TR) \tag{15}$$

where P_{UAV} and P_{MEC} are the power of UAV and MEC core respectively. TR is the total transmission energy for sending the results back to the vehicles. $N'_{UAV}(t)$ and$N'_{MEC}(t)$ are the number of occupied UAV and MEC cores at time t.

3) Penalty: If there are still some tasks unfinished but the UAV $j \in N$ decides to change its position at time t, it will incur a punishment $\phi^j_{U_s}(t)$ for breaking the contract, which is defined as follows:

$$\phi^j_{U_s}(t) = \begin{cases} \kappa \cdot R_{U_s(t)}, \forall j \in N, & \text{if the UAV doesn't} \\ & \text{complete the task } U_s \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

where κ is a punishment coefficient.
The overall penalty of all UAVs over a period of time T is calculated as:

$$P(T) = \sum_{t=0}^{T} \sum_{j \in N} \sum_{s \in V} \phi^j_{U_s}(t) \tag{17}$$

We define the overall cost over a period time T as:

$$C(T) = PC(T) + \delta E(T) + P(T) \tag{18}$$

where δ is the charge per energy unit.

C. Objective

Our goal is to maximize the overall profit over a period time T, which is total revenue minus total cost, and is represented as follow:

$$\begin{aligned} & \max_{d_{s,j}, X_j, Y_j} \quad R(T) - C(T) \\ \text{s.t.} \quad & \text{C1.} \quad \sum_{j=0}^{n+m} d_{s,j} \leq 1, \forall s \in V \\ & \text{C2.} \quad \sum_{i=1}^{v} d_{s,j} \leq N_{UAV}, \forall j \in N \\ & \text{C3.} \quad \sum_{i=1}^{v} d_{s,j} \leq N_{MEC}, \forall j \in M \\ & \text{C4.} \quad T_{Us} \leq T_s^{max}, \forall d_{s,j} = 1 \end{aligned} \tag{19}$$

Algorithm 1 SUM

```
Set the radius Eps and the minimal points Minpts for TSS-DBSCAN.
Set the initial temperature T_0, the cooling rate ρ, and the maximum iterations Iter for SA.
for every timeslot t ∈ [0, T] do
    update the location of each vehicle and check whether there are cores being released
    if t/τ = 0 then
        cluster vehicles with TSS-DBSCAN
        for each UAV j ∈ N do
            if Σ_{s∈V} φ_{Us}(t)^j < ψ then
                UAV j changes to the new location
            end if
        end for
    end if
    for each arriving task U_s(t) do
        build the candidate list according to (2), (7), (8), (9)
    end for
    Generate an initial solution D_0 with greedy algorithm.
    Start SA algorithm with the fitness function (20) until it reaches the setting iteration times and output the best solution D.
    Allocate resources to all the offloading tasks and update the occupied cores for each server.
end for
```

Constraint C1 represents that each task can only be executed at one place. C2, C3 represent that the total CPU cores allocated from a UAV or an MEC server should not exceed the number of cores a UAV or an MEC server has. C4 states that the task execution time should not exceed the delay tolerance.

IV. Proposed Algorithm

We propose an algorithm SUM, which can be divided into two parts. First, we decide the UAV deployment, and then allocate the computational resources to the tasks based on the deployment. The overall procedure of SUM is shown in Algorithm 1.

A. UAV deployment

In this stage, we decide the position of UAVs $\{X_j(t), Y_j(t)\}$ for $j \in N$ based on the distribution of vehicles $\{x_s(t), y_s(t)\}$ for $s \in V$ by TSS-DBSCAN. TSS-DBSCAN is an enhancement clustering algorithm for DBSCAN [21]. It speeds up the execution time by introducing a two-phase filtering technique, avoiding the proliferation of meaningless points. The satellite will first check the location of vehicles and the resource usage of UAVs every timeslot, and decide whether to change their position for receiving much more tasks or not. We set a threshold ψ for the decision to ensure that the incurring break contract punishment won't be too high. If the punishment $\sum_{s \in V} \phi^j_{U_s}(t) > \psi$, the UAV j won't change its position; otherwise, it will change to the location according to the TSS-DBSCAN algorithm. At the end of this step, we will assign UAVs to better locations so that the profit in the next τ timeslots can be increased.

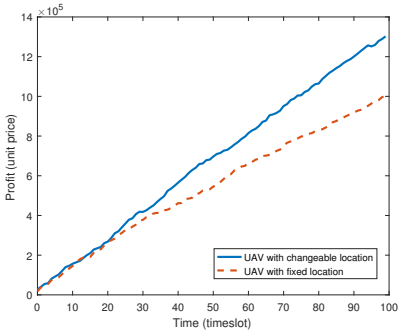

Fig. 3: Comparison between two different UAV deployment schemes

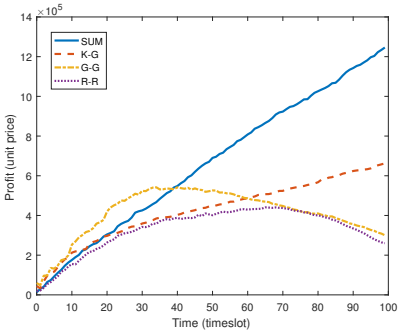

Fig. 4: Accumulated profit over 100 timeslots

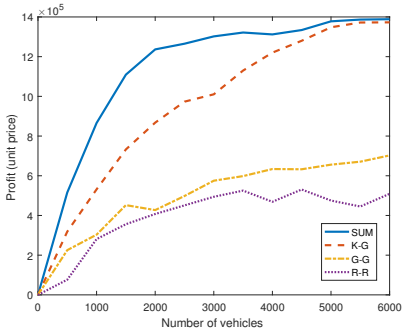

Fig. 5: Profit under different numbers of vehicles

B. Resource allocation

With the locations of UAVs obtained in the first stage, we will then allocate resource to each task according to SA algorithm, which is a technique based on the analogy of metal annealing process for approximating the global optimum [22]. We first set an initial temperature t_0 and the cooing rate ρ, for finding the temperature in the k-th iteration T_k. To ensure that the offloading decision can be made in a reasonable time, we set the maximum iteration for SA algorithm. In addition, we set the fitness function as (20) for evaluating the solutions.

$$\sum_{s \in V} [R_{U_s(t)} - E_{U_s(t)} - PC_{U_s(t)}], \forall t \in [0, T] \qquad (20)$$

In each iteration, SA will generate a potential offloading decision which often referred to "status" based on the previous status. The fitness value of that status can then be calculated. If the fitness value is higher than the previous one, the status will be accepted and replaces the previous status. Otherwise, it will be accepted with the probability of $min\{1, e^{\frac{\Delta f}{T_k}}\}$, where Δf is the difference of the fitness value between the current state and the previous state. SA will output a set of decision variables D with the best fitness value after the iterations.

We first build a candidate list for each arriving task $U_s(t)$ according to (2), (7), (8), (9). Then we generate the initial solution with a greedy algorithm that tends to assign CPU cores with high capacity to those high CPU requirement tasks. After initialization, we start the iteration until it reaches the maximum iteration.

V. Simulation Results

In our simulations, we set the UAV-enabled MEC vehicular system with 1 MEC server and 3 UAVs. Two types of vehicles in our system, 10% for high priority vehicles ($q_s = 1$), and 90 % for low priority vehicles ($q_s = 0$). We assume that all the vehicles are randomly distributed in a 500*500 meter square area with a random velocity, and all of them go forward with a constant speed. Each vehicle generates a task with a probability of 0.05 for every timeslot, and we simulated 100 timeslots for each experiment. Other related parameters are listed in Table 2. The results are derived from the average of 30 simulations.

TABLE II: PARAMETERS SETTING

Parameters	Values	Parameters	Values
N_{MEC}	8	T_s^{max}	$[0.5, 3]s$
N_{UAV}	4	R_{UAV}	$100m$
f_{MEC}	$2100MHz$	R_{BS}	$200m$
f_{UAV}	$1400MHz$	P_{MEC}	$50W$
f_V	$700MHz$	P_{UAV}	$100W$
v_s	$[0, 40]m/s$	η	2000
θ_s	$[0, 2\pi]$	κ	1.2
a_s	$[10, 1000]Kb$	γ_{MEC}	1
c_s	$[20, 2000]Mb$	γ_{UAV}	2

We first evaluate our proposed scheme with UAVs that are fixed at the same places, and the result is shown in Fig. 3. As we can see, UAVs with dynamically adjusted mechanism increases a large amount of profit by about 30%, for the reason that the vehicular distribution might vary as time passes by, and fixed location scheme can not adapt to various scenarios. Even though the fixed location scheme does not incur any penalties mentioned previously, it loses the chance to accept more profitable tasks, and might incur much more potential cost for those unsatisfied tasks.

We further compare our proposed algorithm with the other three existing algorithms, which are K-means with greedy (K-G), greedy with greedy (G-G), and random with random (R-R). K-G classifies vehicles into 3 clusters, and deploys UAVs to the center of each cluster. After the deployment of UAVs, it adopts greedy which tends to allocate resources to the tasks with higher CPU requirements. G-G deploys UAVs by selecting the most 3 popular regions and allocates resources with greedy too. R-R scheme deploys UAVs and allocates resources randomly.

We compare the above 3 methods with our proposed algorithm and observe the accumulated profit over 100 timeslots. The result is shown in Fig. 4. As we can see, R-R has the worst performance because the random algorithm does not assign UAVs to densely populated areas, the CPU cores of UAVs may not be utilized efficiently, and there might be many tasks unserved for lack of resources in popular areas. G-G scheme performs the best at the beginning but declines afterward. We think the reason is that the phenomenon of vehicle cluster is not obvious at the beginning, so greedy might choose better

locations for UAVs. However, since greedy can only select some fixed points for UAVs (we divide the whole 500*500 region into 9 non-overlapping squares for greedy, and the UAVs can only be deployed at the center in each square), once the UAV needs to change its location, the displacement may be larger than K-means or TSS-DBSCAN, which causes the interruption of the services and leads to high penalty. Also, allocate resources with greedy is not good enough since greedy tends to allocate remote servers' cores to those tasks with relatively high CPU resource requirements arrive in the same timeslot. This mechanism makes the cores in remote serves always being occupied. Once there's a task that can not be executed locally due to the delay tolerance and needs to be offloaded, it might not have an idle core to use.

As for K-G scheme which also adopts greedy in the second stage but does not suffer from the profit decreasing, we believe that it's because K-means assigns UAVs to better locations than greedy, which might make up for the deficiency in the second stage. The reason for our proposed algorithm, which adopts TSS-DBSCAN and SA in the first and second stages, respectively, performs better than the other three schemes is that TSS-DBCAN can remove the outliers, so as to get better locations of UAVs. In addition, SA tries different solutions by means of iteration, making a better trade-off between the revenue and the cost.

To better evaluate our proposed algorithm, we applied 12 instances with different numbers of vehicles from 500 to 6000, and 30 independent runs were implemented in each number of instances, as shown in Fig. 5. Our proposed solution earns about twice profit compared with G-G and R-R approaches. The performance of K-G approach becomes closer to our proposed algorithm as the number of vehicles near 5000. It's because with such a large number of vehicles, many tasks compete for the resources at the same time. Even though UAVs select higher profitable tasks, due to the scarcity of the resource, there must still be many unserved tasks which increase the unsatisfied cost, leading to the profit saturated.

VI. Conclusions

In this paper, we proposed an integrated architecture combining a satellite, MEC servers, UAVs to a vehicular network, which may be deployed in the future 6G network. In addition, we proposed a two-stage algorithm SUM for the optimization problem considering the UAV deployment and resource allocation. By dynamically assigning UAVs to proper locations, more tasks can be served, and further increases the profit. The simulation results show that our proposed algorithm achieves better performance over the other approaches. As a future extension of this work, we intend to adopt real traffic flow in our simulations, to better evaluate the performance of our algorithm.

References

[1] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago, "Efficient placement of edge computing devices for vehicular applications in smart cities," *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2018.

[2] U. Lee, R. Cheung, and M. Gerla, "Emerging vehicular applications," *Vehicular networks: From theory to practice*, pp. 6.1–6.30, 2009.

[3] V. Balasubramanian, S. Otoum, M. Aloqaily, I. Al Ridhawi, and Y. Jararweh, "Low-latency vehicular edge: A vehicular infrastructure model for 5g," *Simulation Modelling Practice and Theory*, vol. 98, p. 101968, 2020.

[4] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang, "Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks," *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 288–294, 2016.

[5] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[6] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Task offloading and resource allocation in mobile-edge computing system," *2018 27th Wireless and Optical Communication Conference (WOCC)*, pp. 1–4, 2018.

[7] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Qos-aware mobile edge computing system: Multi-server multi-user scenario," *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2018.

[8] Y. Chiang, Y.-H. Chao, C.-H. Hsu, C.-T. Chou, and H.-Y. Wei, "Virtual network embedding with dynamic speed switching orchestration in fog/edge network," *IEEE Access*, vol. 8, pp. 84 753–84 768, 2020.

[9] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2017.

[10] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: Design, deployment and performance analysis," *2015 IEEE global communications conference (GLOBECOM)*, pp. 1–6, 2015.

[11] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.

[12] X. Diao, J. Zheng, Y. Wu, Y. Cai, and A. Anpalagan, "Joint trajectory design, task data, and computing resource allocations for noma-based and uav-assisted mobile edge computing," *IEEE Access*, vol. 7, pp. 117 448–117 459, 2019.

[13] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, 2019.

[14] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[15] S. W. Loke, "The internet of flying-things: Opportunities and challenges with airborne fog computing and mobile cloud in the clouds," *arXiv preprint arXiv:1507.04492*, 2015.

[16] N. Tafintsev, M. Gerasimenko, D. Moltchanov, M. Akdeniz, S.-P. Yeh, N. Himayat, S. Andreev, Y. Koucheryavy, and M. Valkama, "Improved network coverage with adaptive navigation of mmwave-based drone-cells," *2018 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–7, 2018.

[17] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48–54, 2018.

[18] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061–3074, 2019.

[19] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2018.

[20] G. Wu, Y. Miao, Y. Zhang, and A. Barnawi, "Energy efficient for uav-enabled mobile edge computing networks: Intelligent task prediction and offloading," *Computer Communications*, vol. 150, pp. 556–562, 2020.

[21] C.-F. Tsai and Y. Chiang, "Enhancement of data clustering using tss-dbscan approach for data mining," *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 2, pp. 535–540, 2016.

[22] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," *Simulated annealing: Theory and applications*, pp. 7–15, 1987.