Vehicular Computation Offloading for Industrial Mobile Edge Computing

***Abstract*—Due to the limited local computation resource, industrial vehicular computation requires offloading the computation tasks with time-delay sensitive and complex demands to other intelligent devices (IDs) once the data is sensed and collected collaboratively. This paper considers offloading partial computation tasks of the industrial vehicles (IVs) to multiple available IDs of the industrial mobile edge computing (MEC), including unmanned aerial vehicles (UAVs), and the fixed-position MEC servers, to optimize the system cost including execution time, energy consumption, and the ID rental price. Moreover, to increase the access probability of IV by the UAVs, the geographical area is divided into small partitions and schedule the UAVs regarding the regional IV density dynamically. A minimum incremental task allocation (MITA) algorithm is proposed to divide the whole task and assign the divided units for the minimum cost increment each time. Experimental results show the proposed solution can significantly reduce the system cost.**

***Index Terms*—Mobile edge computing, task allocation, unmanned aerial vehicles, game theory, industrial vehicular computation offloading.**

I. Introduction

THE demand for the internet of things (IoTs) in industrial production has been increasing along with the rapid development of the industrial IoTs [1]. In the production site, with vehicular networks (VNs), industrial vehicles (IVs) need to execute the computation tasks of applications to achieve intelligent manufacturing including data collection and analysis, on-site safety exploration, obstacle early warning, and task dispatch. These applications are often time-delay sensitive and complex, where high latency can cause unforeseen consequences, and even affect the safety of the operator's life. However, at present, it will be very costly to equip powerful computing device for a large number of IVs. As a result, most of the IV computing capacity is not enough to complete complex applications within the maximum latency.

Computation offloading is one of the key technologies to solve the issues stated above. For vehicular computation offloading, the IVs without enough computing power can offload all or part of their computation tasks to other intelligent devices (IDs) through wireless access [2], [3], [4]. Then, these IDs will work on behalf of the vehicle to execute computation tasks and return the results to the requesting vehicle. In addition, global information is needed to make a decision for vehicle computation offloading while the interconnection of global network depends on VNs. However, VNs are easily affected by dynamic topology changes, which will lead to the instability of IVs wireless links. This cannot guarantee the stability of the information exchange. The emergence of software-defined network (SDN) architecture opens up new doors to solve the issue of VNs [5], [6]. SDN architecture separates the control plane and data plane to realize flexible and intelligent control of the network. The SDN controller can collect the data uploaded by each vehicle and the IDs in real-time, such as location and speed to realize the interconnection of the global network. Moreover, mobile edge computing (MEC) can provide users with a computing platform with strong computing power and fast response through wireless access [7]. However, due to the high construction cost, the temporary industrial production sites are with less infrastructure in most remote areas; the signal of IVs transmitted may be blocked by the obstacles for instance buildings, which will lead to weak transmission signals or even interruption. All these factors lead to the enormous challenges of offloading the computation task of IVs through wireless communication.

Unmanned aerial vehicle (UAV) can solve the problems stated above with its advantages of small size, simple deployment, and low price [8]. In the industrial production site, deploying multiple UAVs with strong computing power can provide computing services for IVs. As all services are priced at present, IVs need to pay a certain fee to offload computation tasks to UAVs or MEC to hire them to execute computation tasks instead of themselves. Hence, it is a key challenge for the whole system to minimize the cost (i.e., execution time, energy consumption and rental price) of executing the computation tasks of IVs.

In this paper, we formulate the optimization of computation task offloading cost of IVs based on SDN architecture in the temporary production site with less infrastructure. The SDN controller offers the global IVs to obtain information regarding the offloading computation task. Multiple IDs are deployed in the system, including multiple UAVs and a MEC server to provide computing services for IVs. An IV can offload its computation task to multiple IDs that can establish a wireless connection. Then, the decision-making problem of multi-IV computation task offloading as a multi-device for multi-user sequential game and prove the existence of Nash equilibrium is formulated. With the objective of jointly optimizing the execution time, energy consumption and price of computation tasks, we designed the minimum incremental task allocation (MITA) algorithm to dynamically assign the computation tasks of IVs to multiple IDs that can establish wireless communication. Different from the existing work [2], to save local computing resources, we consider offloading part of the computation task of IV. Moreover, different from [9], we offload the computation task to multiple available IDs, in which multiple IDs and local parallel execution of computation task can greatly reduce the execution time of the task. The main contributions are summarized as follows.

- **Multi-device for multi-user computation offloading game**. IVs can assign computation tasks to multiple available IDs to execute the task in parallel, which

can greatly improve the computing efficiency. This is unlike [9]. Moreover, we formulate the multiple-target computation offloading for multiple IVs as a multi-device for multi-user computation offloading sequential game, by optimizing the system cost of performing computation tasks to develop the optimal offloading scheme. Here, we consider that the system cost is mainly composed of execution time, energy consumption and rental IDs price.

- **Vehicle density-based UAV dispatch**. In order to promote the probability of establishing wireless links between IVs and UAVs, a dynamic scheduling scheme for UAVs based on the IVs density of small partitions is introduced. The residence time of the UAV in the destination partition scheduled by the SDN controller depends on the current density of vehicles in the partition.
- **Minimum incremental task allocation (MITA) algorithm**. We formulate a task allocation algorithm for IVs to offload part of the computation tasks to multiple IDs. MITA can minimize the system cost of executing tasks and work out the optimal multi-target task allocation scheme.

The rest of this paper is organized as follows. In Section III, we introduce the system scenario. We next formulate the problem in Section IV. Then, Section V introduces our proposed MITA algorithm. The evaluation results are shown in Section VI. Finally, Section VI concludes the paper.

II. Related Work

In the recent years, the existing work of the SDN assisted VNs emerges in an endless stream [5], [6], [10], [11]. Within the SDN framework, the controller updates the data in real-time, adjusts and schedules the network in a centralized manner to improve the performance of the VNs. In addition, large number of emerging VN applications not only provide services in terms of security, also entertainment services [12],which can enrich people's travel experience. These network applications usually demand high computational load and time-delay sensitive. Therefore, the vehicle needs to consider offloading all or part of the complex computation tasks to other IDs via the wireless medium to perform the tasks instead of itself. The existing studies of vehicle computation offloading is generally offload vehicular computation tasks to the cloud via wireless links, where cloud computing can supply efficient computing services for vehicles, for instance in [2], [4]. However, most of the base stations that provide cloud computing services are far away from users, in which offloading computation tasks will increase the cost of energy and latency. In this context, the MEC server can provide computing and caching services to users nearby to improve response speed and service quality [7]. Some other work [3], [13], [14] considers offloading the computation tasks of mobile users to the MEC server to provide users with efficient computing services. In addition, in [15], [16], [17], the authors combine the cloud computing with MEC to provide computing services for mobile users. In [15], considering the remote cloud computing capability is stronger than the edge cloud with a far distance, rational use of the computing resources of these two devices is applied to minimize energy consumption. Wang et al. introduce the real-time traffic management of the fog-based Internet of Vehicles (IoV) system in [18] to optimize the offloading scheme by minimizing the average response time of offloading. However, transferring the entire computation task to another device will result in the waste of local computing resources and data transmission energy of the vehicle. Moreover, the wireless communication link of the vehicle is easily affected by dense buildings and various interference signals, which can result in a reduction in transmission rate or even communication link interruption. This will affect the efficiency of computation offloading. Therefore, deploying UAVs to assist the computation offloading is one of the effective solutions to this problem.

Recently, UAVs have been widely deployed in various fields, in particular used to assist ground communication and computing. For instance, as described in [8], UAVs are applied in civilian due to the nature of low price, easy deployment, and ignorance of terrain. In some other studies [19], [20], in order to expand the communication range of ground nodes and improve the quality of communication services, UAVs are also deployed as air base stations. Specially, in [21], UAVs can also assist the MEC server to compute tasks and cache data by alleviating the resource scheduling problem of ground nodes. In order to reduce the workload of the vehicle and improve the efficiency of data processing, in [22], UAVs are used as a relay nodes to assist the vehicle when forwarding data to the MEC server. Therefore, the deployment of UAVs can help to solve a variety of ground communication and computing problems, while deployment of UAVs-assisted vehicles is one of the solutions to realize intelligent transportation.

Consequently, how to allocate resources in the system reasonably and efficiently has become a challenging issue. [9], [23] offload partial computation tasks to other devices, which improve the service quality by optimizing resource allocation, reducing delay and energy consumption. Since the total resources of the system are limited, how to operate scheduling and allocation dynamically according to the overall demands is the key to enable users to deal with various complex applications efficiently. In [24], an adaptive resource management scheme is proposed where the cost of resource management is greatly reduced without performance degradation. In [25], Zhang et al. propose a three-layer computing offloading environment, in which the resource allocation problems such as partial offloading and wireless resource scheduling are considered with the goal of optimizing energy consumption. The IDs in the system provide users with computing, caching, and other services, in which users need to pay corresponding fees for various services provided by suppliers. In [26], authors propose a unified pricing scheme according to the different degree of sparsity of user distribution, and schedules resources on basis of the price and transmission power. [27] studies the pricing and power distribution of the spectrum network, finds the equilibrium point of the solution by adopting the game theory, and enables the user to obtain the optimal profit. On the other hand, [28] focus on the optimization of energy distribution of UAV, and dynamically price the service of UAV according to the remaining resources and hover time. To sum up, most existing studies [2], [5] of computation task offloading aim at

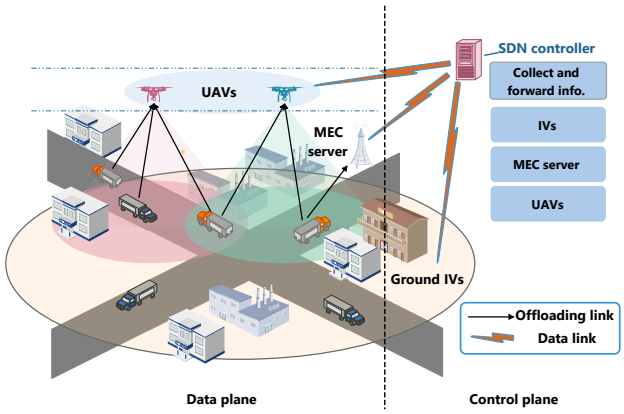

Fig. 1. System Model

optimizing execution time and energy consumption only. To be more realistic, this paper counts the price of renting IDs to provide computing services as one of the costs of performing tasks.

Different from existing work [2], [9], we consider offloading partial computation task of IV to multiple IDs at the same time. In addition, we adopt the method of sequential game theory to make the offloading decision. The optimal task allocation algorithm, MITA, is designed by jointly optimizing the execution time, energy consumption, and the price of computing services provided by IDs. Therefore, IVs can independently work out the optimal task allocation scheme.

III. System Model

This section introduces the system model of IVs offloading computation tasks as shown in Fig. 1. We set up a set $N = \{1, 2, 3, ..., n\}$ to represent the IVs driving in the temporary industrial production site who generated a set of computing complex and time-delay sensitive computation task A. There is only one MEC server in the area that can provide the service of offloading computation for the IVs in which this MEC server is with strong computing capability. However, there are some buildings in the scenario, which may block the communication signal between the IVs and MEC server. In addition, as the MEC server is far away from the IVs, it can consume a lot of energy to offload the computation tasks through wireless transmission. Thus, we deploy multiple UAVs with certain computational power to provide computing services for IVs. We consider a set of $M = \{1, 2, 3, ..., m\}$ to represent the UAVs in the region. The interference of wireless communication signal increases due to the possibility of vehicle density in some small areas. Hence, we schedule UAVs according to the vehicle density of each small area. Then, more UAVs are dispatched to the area with higher IV density to provide services for this partition. We design an IVs computation offloading architecture based on SDN. The controller of SDN is applied to collect and forward the relevant information of devices in the system, so as to reduce the communication pressure if we use the vehicle to collect the information. Moreover, different from the static scenario of existing work [2], we intend to make our scenario more realistic by allowing the vehicles and UAVs moving. As the computation model and communication model are the key factors of the task offloading, we will introduce the two models in the following.

A. Communication Model

We consider two schemes by having the scheme set as $K = \{0, 1\}$, where $k_n = 0$ represents that IV_n cannot establish a wireless link with any other device, but can only execute the computation task locally; $k_n = 1$ denotes that IV_n can establish wireless links with other IDs and offload the computation tasks to these devices to execute instead of itself. According to the scheme set K, we can give the transmission rate of the IV_n offloading computation task A to the target device D through wireless communication as

$$r_n(D) = W log_2(1 + \frac{P_n H_{n,D}}{N_0 + \sum_{i \in N, i \neq n} P_n H_{i,D}}) \quad (1)$$

where W is the channel bandwidth, and P_n is the transmission power of the IVs; N_0 is the background noise power; $H_{n,D} = g h_{n,D}^{-\partial_V}$ is the channel gain between the IV_n and target device D (i.e., UAVs or MEC server) in which g is the fading component, and $d_{n,D}$ is the distance between the IV_n and target D; $\sum_{i \in N, i \neq n} P_n H_{i,D}$ means the IVs (excluding IV_n) send data to the same target D via wireless access simultaneously. We can observe it from (1), the more other IVs (excluding IV_n) send data to the target D via wireless access at the same time, which will cause severe interference and the reduction of transmission rate.

B. Computation Model

IVs can generate computation tasks in real-time. Here, let $A_n = (B_n, D_n, J_n), A_n \in A$ to represent the computation task generated by the IV_n. B_n denotes the number of Central Processing Unit (CPU) cycles needed for executing computation task A_n; D_n denotes the size of offloading computation task A_n data; J_n means the returned result size of the computation task A_n.

In the scenario, the IVs are not equipped with strong computing power which means they cannot complete executing complex and time-delay sensitive computation tasks within the maximum time-delay. Therefore, when a IV generates a computation-intensive and delay-sensitive task that needs to be executed, the IV will consider offloading the partial computation task to other IDs via wireless links. In this paper, IVs can offload computation tasks to UAVs and a MEC server if they can establish wireless connections. However, as the total resources of the system are limited, it is a key challenge to make IVs adopt these resources reasonably with minimized system costs. Next, we will introduce the computation model according to different schemes.

1) Local computing: In one circumstance, IV_n cannot establish a communication connection with any other ID. Consequently, IV_n needs to execute all computation task A_n locally. In this way, the execution time of the computation task A_n depends on the computing power (F_{cpu}^V) of the IV_n, that

is, the cycles per second of the CPU. Hence, we can simply calculate the execution time as

$$T_n^L = \frac{B_n}{F_{cpu}^V} \tag{2}$$

Moreover, the energy consumption of IV_n for executing the computation task A_n locally depends on the energy consumption per CPU cycles (L_{cpu}^V). Then, we can give that the energy consumption of IV_n for executing computation task A_n locally as

$$E_n^L = B_n L_{cpu}^V \tag{3}$$

2) Computation offloading: In the case of IV_n accessible to other IDs, the IDs that can provide computing services for IV_n containing a MEC server and multiple UAVs. In addition, we also set a waiting queue for each ID, which is used to store computation tasks and make full use of the resources of the ID. When a complex and time-delay sensitive computation task generated by IV_n needs to be executed at the current time, IV_n will first broadcast the request signal to request the computing service. Then, the ID that can receive the request signal will send a signal that can be offloaded to the IV_n. Thus, we set $Q = \{1, 2, 3, ..., m, m+1\}, m \in M$ to indicate the number of the MEC server and UAVs that can establish wireless links with IV_n. In order to fully utilize the resources of all devices in the system and improve the throughput(i.e., the number of computation tasks executed in the system per minute), we enable the IV_n to offload a portion of computation task A_n to the other IDs with the wireless connection, that is, execute task A_n locally in parallel with these available IDs. The execution time of the offloading part (T_n^{off}) is the maximum value of the execution time of all the IDs providing computing services for IV_n. Hence, the total execution time of computation task A_n is the combination of the maximum value of the local execution time (T_n^{loc}) and the execution time of the offloading part (T_n^{off}). In addition, we set $X_Q = \{x_1, x_2, ..., x_q\}$, where each value of this set represents the percentage of computation task A_n that ID is offloaded. In addition, x_q represents the percentage of computation task A_n offloaded to the MEC server, in which the remaining part of the task is offloaded to UAVs. Then the total execution time of the offloading scheme of computation task A_n is

$$T_n^O = max(T_n^{loc}, T_n^{off}) \tag{4}$$

where

$$T_n^{loc} = \frac{(1 - \sum_{i=1}^{q} x_i) B_n}{F_{cpu}^V} \tag{5}$$

$$T_n^{off} = \max_{0<q\leq m+1} \left\{ T_{off}^1, T_{off}^2, ..., T_{off}^q \right\} \tag{6}$$

and

$$T_{off}^1 = \frac{x_1 D_n}{r_n(1)} + \frac{x_1 B_n}{F_{cpu}^{UAV}} + \frac{x_1 J_n}{r_n(1)} + T_{wait}^1 \tag{7}$$

$$T_{off}^2 = \frac{x_2 D_n}{r_n(2)} + \frac{x_2 B_n}{F_{cpu}^{UAV}} + \frac{x_2 J_n}{r_n(2)} + T_{wait}^2 \tag{8}$$

...

$$T_{off}^{q-1} = \frac{x_{q-1} D_n}{r_n(q-1)} + \frac{x_{q-1} B_n}{F_{cpu}^{UAV}} + \frac{x_{q-1} J_n}{r_n(q-1)} + T_{wait}^{q-1} \tag{9}$$

$$T_{off}^q = \frac{x_q D_n}{r_n(q)} + \frac{x_q B_n}{F_{cpu}^{MEC}} + \frac{x_q J_n}{r_n(q)} + T_{wait}^q \tag{10}$$

where T_{off}^1, T_{off}^2 and T_{off}^{q-1} represent the execution time of the partial computation task A_n executed by UAV_1, UAV_2 and UAV_{q-1} respectively; Moreover, if IV_n offloads part of the computation task A_n to multiple UAVs that can establish communication links, the task execution time of the UAV can be calculated according to equation (7), (8) and (9); T_{off}^q denotes the execution time of the partial computation task A_n executed by MEC server; F_{cpu}^{MEC} represents the computing capability of the MEC server; T_{wait}^1, T_{wait}^2 and T_{wait}^q indicate the current waiting queue length of UAV_1, UAV_2 and UAV_{q-1}, respectively; T_{wait}^q is the waiting queue length of MEC server; (T_n^{loc}) and (T_n^{off}) represent the execution time of the local execution part and the execution time of the offloading part, respectively. In addition, if the vehicle cannot establish a communication link with a certain ID, the percentage of task A_n allocated to the device is $x_i = 0$. Since all the IDs providing computing services for IV_n are parallel computing, the longest execution time of the offloaded IDs is defined as the total execution time of the offloading part. Next, we can give the energy consumption of executing computation task A_n in the computation offloading scheme (E_n^O) as the sum of the energy consumption of the local part (E_n^{loc}) and the energy consumption of the offloading part (E_n^{off}). We have that

$$E_n^O = E_n^{loc} + E_n^{off} \tag{11}$$

where

$$E_n^{loc} = (1 - \sum_{i=1}^{q} x_i) B_n * L_{cpu}^V \tag{12}$$

and

$$E_n^{off} = E_{off}^1 + E_{off}^2 + ... + E_{off}^q \tag{13}$$

where E_{off}^1 to E_{off}^{q-1} denotes the energy consumption of executing part of the computation task A_n of UAV_1 to UAV_{q-1}; E_{off}^q represents the energy consumption of the computation task A_n of MEC server. We give details as follow

$$E_n^{off} = x_q D_n L_{send}^V + \sum_{i=1}^{q-1} x_i B_n L_{cpu}^{UAV} + \sum_{i=1}^{q-1} x_i D_n L_{send}^V + \sum_{i=1}^{q-1} x_i J_n L_{send}^{UAV} \tag{14}$$

The rental price is one cost component of the computation task offloading. Computation task offloading is a service in which other IDs lease the computing resources to IVs to improve computing efficiency. However, a certain fee is required for rental of computing services, i.e., the IV needs to pay a certain amount for each CPU cycle of rented ID. In this paper, we define that the price of computing service supplied by UAVs and the MEC server is not the same, the price of each one CPU cycle that IVs hire from UAVs and MEC server is p_{cpu}^{uav} and p_{cpu}^{mec} respectively. Then, the total price of offloading computation task A_n scheme can be given as

$$P_n^O = x_q B_n * p_{cpu}^{mec} + \sum_{i=1}^{q-1} x_i B_n * p_{cpu}^{uav} \tag{15}$$

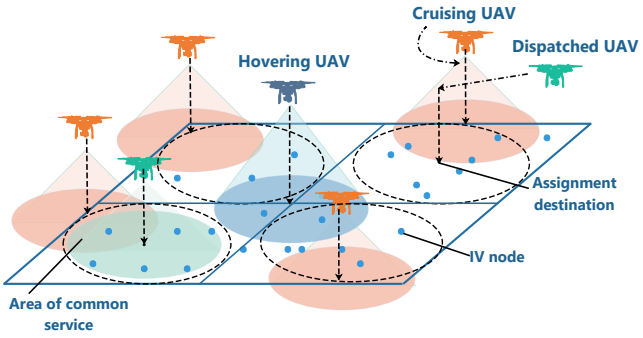

Fig. 2. The UAV trajectory model

C. Payoff Function

The cost of the system mainly consists of execution time, energy consumption and rental price. The payoff function is used to measure the cost of executing computation task generated by the IV. We can then calculate the total cost of the performing computation task as follows

$$C_n = \alpha T_n + \beta E_n + \gamma P_n \tag{16}$$

where α, β, and γ represent the weight of execution time, energy consumption, and rental price of the computation task in the cost, respectively, where $\alpha + \beta + \gamma = 1$. The weight represents the proportion of the cost factor in the total utility. For instance, the higher the weight of execution time is, the better the scheme with shorter execution time is when selecting the optimal scheme. Because of the different requirements of the computation tasks, each task is set with weight values, including execution time, energy consumption and rental price. In the case of local execution, the price cost of renting IDs is not involved. Therefore, according to (2) and (3), the total cost of the IV_n executing computation task A_n locally can be calculated as

$$C_n^L = \alpha T_n^L + \beta E_n^L \tag{17}$$

and for the payoff function of the offloading computation scheme, the cost of renting computing resources from other IDs is considered. We have that

$$C_n^O = \alpha T_n^O + \beta E_n^O + \gamma P_n^O \tag{18}$$

In particular, if the user desires to reduce the cost of the price, the user can increase the weight of rental price to achieve the expected objective; If the computation task generated by the user is time-delay sensitive, the weight of execution time can be increased appropriately.

D. The UAV Trajectory Model

In this subsection, the trajectory of the UAVs will be presented. The trajectory of UAVs can be divided into two types, where the first is with a fixed trajectory, the other is dynamic scheduling based on regional vehicle density. First, we divide the whole topology into four equally sized partitions. For the fixed trajectory, we deploy four UAVs to cruise at a constant speed on a fixed trajectory in each partition and deployed a UAV that hovered at the center point of the entire topology. For dynamic scheduling UAVs, except for five UAVs with fixed trajectory, other UAVs conduct dynamic scheduling according to the regional vehicle density of each partition provided by the SDN controller. The higher the regional vehicle density in the current partition, the more likely the UAV will be dispatched to this partition. When the UAV needs to be dispatched to a certain partition, the destination of this dispatch is set as the midpoint of the partition. As the UAV reaches the destination, the hovering time is determined by the regional vehicle density of the partition at that time. Until the end of hover, the UAV will accept the new scheduling again. The trajectory model of UAVs in the system is shown in Fig. 2 with illustrating various moving modes of UAV. Here, the assigned UAVs are briefly shown in the figure. The number of UAVs to be assigned depends on the total number of UAVs deployed in the system (i.e., the total number minus 5).

IV. Multi-Device for Multi-User Sequential Game

In this section, we formulate the scenario in which multiple IDs provide computing services. Each IV performs local computing or offloading part of the computation task as a multi-device for multi-user sequential game. The main objective is to optimize the system cost of executing vehicular computation tasks, including the execution time, energy consumption, and rental price. By studying the existence of Nash equilibrium point in this game, we can find the optimal utility of each user in the system, and then we can make the mutually satisfactory offloading decision.

Game theory is to study the interaction between participants and design game mechanism to get the optimal decision. Through the game mechanism, we can design the computation task offloading mechanism to get mutually satisfactory benefits. Hence, no participant has the motivation of private deviation. Game theory is a typical idea that can devise a distributed mechanism, in which each participant can plan their own satisfactory offloading scheme. This can reduce the computational pressure of centralized management and scheduling of central controller.

A. Game Mechanism

We formulate the decision-making problem of multi-device service for multi-user computation task offloading as multi-device for multi-user sequential game. Each user makes the offloading decision in a certain order. Users should observe the actions of users who have made previous decisions, then make relevant decisions. By deploying the SDN controller to collect and send global information to users in the system, users can make decisions under the condition of complete information (i.e., waiting queue length, geographic location, and CPU computing power of available IDs) and improve the efficiency of IV decision-making. The multi-device for multi-user sequential game can be defined by $G(N, K, U)$ and the three elements of the game can be given as

1) $N = \{1, 2, 3, ..., n, ..., e\}$, represents the vehicle players who produce time-delay sensitive and complex computation tasks in the game.

2) $K_e = \{k_1, k_2, ..., k_n, ..., k_e\}$, means the strategy set of vehicle players, where strategy $k_n = 0$ represents the vehicle n chooses to execute the computation task locally, and $k_n = 1$ represents that vehicle n selects to offload part of the computation task to other IDs to perform this part of the computation task instead of the vehicle n. These IDs can establish a wireless connection with vehicle n , including UAVs and MEC server.
3) Utility function, U_n represents a function that measures the cost when vehicle player n performs a computation task. The cost mainly includes the execution time, energy consumption and the price of renting IDs. The utility function of the player n $k_n(k_n \in K_e)$ can be given as

$$U_n^{k_n} = (1 - k_n) * C_n^L + k_n * C_n^O \tag{19}$$

where we consider the offloading decision of vehicle player n is $k_n \in \{0, 1\}$. In addition, the computation offloading decision of other vehicle players except player n is $k_{-n} = (k_1, ..., k_{n-1}, k_{n+1}, ..., k_e)$. Hence, player n intends to choose a strategy to minimize the cost of execution time, energy consumption and rental price of executing computation task, which can be expressed as

$$\min_{k_n \in \{0,1\}} U_n(k_n, k_{-n}), \forall n \in N$$

According to (15), we can give the cost of user n as follows

$$U_n(k_n, k_{-n}) = \begin{cases} C_n^L, & if\ k_n = 0, \\ C_n^O, & if\ k_n = 1. \end{cases} \tag{20}$$

Next, we will investigate the existence of the Nash equilibrium (NE).

Definition 1. A strategy profile $K_e^* = (k_1^*, k_2^*, ..., k_n^*, ..., k_e^*)$ for the sequential game $G(N, K, U)$ and if

$$U_n(k_n^*, k_{-n}^*) \leq U_n(k_n, k_{-n}^*), \forall k_n \in K_e \tag{21}$$

then the strategy profile K_e^* is the NE of the game G. When at the NE, no player can increase its profit by deviating from this strategy.

We can conclude that, when the game G is at the NE, each IV makes the optimal decision to minimize its computing cost. Next, we will study the optimal strategy of IVs.

B. Best Strategy for Users

For IV_n, in order to maximize its own profits , user n needs to solve the following problems to get the optimal strategy k_n^*.

$$\mu_n(k_n) = arg \min_{\{T_n, E_n, P_n\}} U_n^{k_n} = (1 - k_n) * C_n^L + k_n * C_n^O \tag{22}$$

According to (22), it can be easily proved that the formula for solving the optimal strategy is convex.

$$\begin{aligned} \frac{\partial^2 U_n^{k_n}(T_n, E_n, P_n)}{\partial^2 T_n} &= 0, \\ \frac{\partial^2 U_n^{k_n}(T_n, E_n, P_n)}{\partial^2 E_n} &= 0, \\ \frac{\partial^2 U_n^{k_n}(T_n, E_n, P_n)}{\partial^2 P_n} &= 0 \end{aligned} \tag{23}$$

From the previous equations, we can conclude that (22) is a convex equation. Hence, there is an optimal strategy, and it exists NE in the multi-device service for multi-user sequence game. Then, the optimal strategy k_n^* of IV_n can be given as

$$k_n^* = \mu_n(k_n) = \begin{cases} arg \min_{\{T_n, E_n, P_n\}} C_n^L & if\ k_n = 0 \\ arg \min_{\{T_n, E_n, P_n\}} C_n^O & if\ k_n = 1 \end{cases} \tag{24}$$

Algorithm 1: Calculation Cost Increment

Input: The parameters of computation task $A_n = (B_n, D_n, J_n)$ generated by IV_n, including the number of CPU cycles needed for computation task A_n to be executed (B_n); the size of offloading computation task A_n data (D_n); the result size of the computation task A_n (J_n), Compute T_{wait}, $T_{assigned}$ and T_{max};

Output: Allocate unit size computation task A_n to device I, the incremental size of system cost.

Initialization: Available device I executes unit task A_n, the system cost increment $\Delta u = 0$, time increment $\Delta t = 0$ and the increment of energy consumption and price $\Delta ep = 0$.
Compute the execution time T_{unit} of unit task A_n, U_{unit}^{ep} the weighted sum of cost in addition to time (i.e., energy consumption and price).
if $T_{max} < T_{wait} + T_{assigned} + T_{unit}$ **then**
  set Δt is the utility of $T_{wait} + T_{assigned} + T_{unit} - T_{max}$
  set Δep is the utility of U_{unit}^{ep}
  set $\Delta u = \Delta t + \Delta ep$
  return Δu
end
else
  set $\Delta u = \Delta ep$
  return Δu
end

V. Computation Task Offloading Allocation Mechanism

In this section, we propose a minimum incremental task allocation (MITA) algorithm to ensure the efficient execution of vehicular computation task, keeping the system cost to a minimum. Next, we divide it into two steps to get the optimal offloading decision, and introduce the design of the mechanism in detail.

A. Calculate Cost Increment

In this paper, offloading vehicular computation task to multiple IDs, including UAVs and MEC server with communication connections is considered. Our proposed algorithm is to minimize the system cost of executing computation task by minimizing the utility increment of the unit task. Here, we will introduce the method of assigning unit size computation task to the available devices and calculating system utility increment. Therefore, when a computation task A_n generated by the IV_n needs to be executed, the IV first broadcasts the offloading signal to determine which IDs can establish

Algorithm 2: Minimum Incremental Task Allocation

Initialization: $\varphi = 0$.
while $\varphi < 1$ **do**
  Compute the utility incremental Δu of each available device to perform unit computation task A_n according to Algorithm 1
  Compare the incremental size of each available device $\Delta u_1, ..., \Delta u_i$ to execute unit task A_n
  Allocation computation task A_n of unit size for the device with the smallest utility increment
  Set the value of φ plus the computation task A_n of unit size allocated this time
end

wireless links with themselves. After that, the IV should decide how to assign the computation task to multiple available devices. Hence, we design this algorithm to calculate the utility increment (i.e., cost increment) of each available device. Next, the process of assigning computation task A_n generated by IV_n is represented.

The IV_n first divides computation task A_n into small parts (a_n) of the unit size, then performs Algorithm 1 to calculate the system cost increment of task a_n of each available device, and assigns the computation task of the a_n size to the device I with the smallest system increment. Here, the devices contain the available IDs and the vehicle itself. Before performing Algorithm 1, the IVn needs to calculate the waiting time T_{wait} of the waiting queue of device I; the execution time $T_{assigned}$ for part of task A_n previously assigned to device I and the current maximum time T_{max} (i.e., waiting time and execution time of assigned part of task A_n) of each available device of IV_n. After these parameters are calculated, Algorithm 1 is performed to calculate the utility increment of unit task a_n to device I. The calculation cost increment is given in Algorithm 1.

According to Algorithm 1, the increment system cost of each available device for executing unit task a_n can be calculated. Next, we will allocate computation task A_n according to the calculated utility increment of each available device.

B. Minimum Incremental Task Allocation Algorithm

In this subsection, we represent the main process of assigning computation task A_n generated by IV_n to each available device. In addition, we define the percentage of task A_n that has been assigned as φ. Then, the MITA algorithm is described in Algorithm 2.

We propose the MITA algorithm as shown in Algorithm 2. Here, we can solve the problem of allocating tasks to multiple devices with the motivation of minimum system cost. Multiple IVs generate the time-delay sensitive and complex computation tasks in the system with less infrastructure (only one MEC server). Each IV will independently make a task execution decision that is mutually satisfactory with other users. To synchronize the clock signal, we use the clock signal from wireless access MEC server for synchronization. The algorithm ends when all users' tasks have been completed and achieved NE.

TABLE I
MAIN PARAMETERS

Parameters	Description	Value
B_n	Computation cycles for task A_n (Megacycles)	[4000,...,5000
D_n	Offload data size of task A_n (K bytes)	[2000,..., 25000]
J_n	Results data size of task A_n (K bytes)	[50,..., 150]
W	Wireless channel bandwidth (MHz)	20
P_n	Transmission power of vehicle (mWatts)	100
R_{uav}	Communication radius of UAV (m)	400
F_{cpu}^{V}	CPU computing power of IV (GHz)	0.7
F_{cpu}^{UAV}	CPU computing power of UAV (GHz)	10
F_{cpu}^{MEC}	CPU computing power of MEC server (GHz)	70
L_{cpu}^{V}	Energy consumption per CPU cycles of IV (J/Megacycles)	1
L_{cpu}^{UAV}	Energy consumption per CPU cycles of UAV (J/Megacycles)	1
L_{send}^{V}	Energy consumption of IV send one data unit through wireless channel (J/KB)	20
L_{send}^{UAV}	Energy consumption of UAV send one data unit through wireless channel (J/KB)	20
p_{cpu}^{mec}	Price per CPU cycles of MEC server (Megacycles)	10^{-5}
p_{cpu}^{uav}	Price per CPU cycles of UAV (Megacycles)	3×10^{-5}

VI. NUMERICAL RESULTS

A. Configuration

In this section, we evaluate the performance of MITA algorithm to offload IV computation task to multiple IDs in terms of optimizing system energy consumption and improving system throughput. In order to simulate a more realistic vehicular computation offloading scenario, we use Java to build a multi-device assisted multi-vehicle computation offloading platform which has been made open-source (Link: https://github.com/ykqykq/MITA). The platform includes communication module, computing module, and mobility module. These modules are executed in parallel by multiple threads (i.e., UAV threads, the MEC server thread, and vehicle threads) to implement mobile computation offloading. Moreover, We adopt a 2000m $\times$ 2000m topology and use the traffic simulation software SUMO (Simulation of Urban MObility) to simulate the trajectory of multiple vehicles. The CPU computing power of MEC server is considered ten times powerful than that of UAV. However, the UAVs are close to the user which also enable IVs to be served faster. In addition, the computing power of IVs is poor, and the CPU computing power of MEC server is 100 times higher than that of IVs. This study also considers that the energy consumed by vehicle and UAV to transmit unit size data through wireless access is 40 times more than the local computation energy cycle of the same data unit on the vehicle and UAV. As for the price, the price per CPU cycle of UAV is three times that of MEC server. However, the MEC server is far away from users and the wireless link may be blocked by obstacles, in this case, only UAVs can provide computing services for users. The simulation parameters of this paper are shown in Table I by considering the experimental parameters of related work[2], [29].

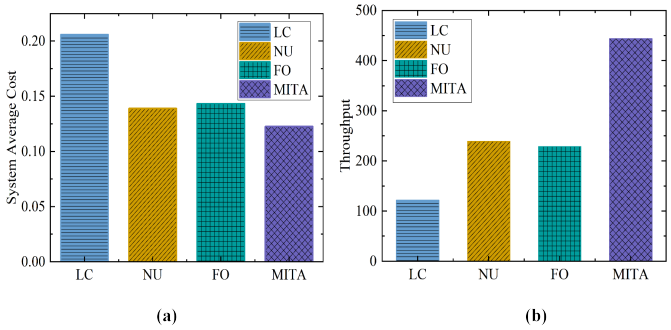

Fig. 3. System average cost and throughput in different scenarios

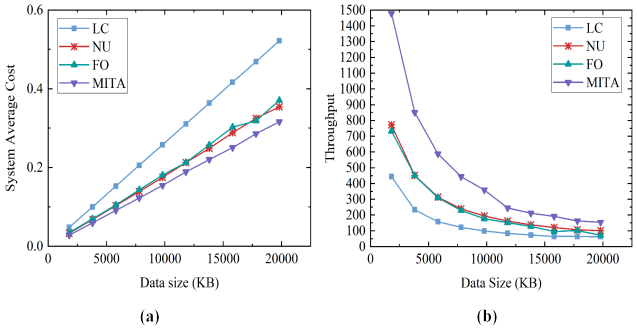

Fig. 4. System average cost and throughput under the change of computation task size generated by IVs

B. Performance

In this part, we simulate the performance of the MITA algorithm based on the sequential game and several other scenarios under the same or different parameters to show the usability of our algorithm. The other scenarios include: 1) All IVs execute computation tasks locally (Local computing, LC); 2) Only a MEC server in the system can provide computing services for IVs while there is no UAV in the system (No UAV, NU); (3) IVs offload all their computation tasks to other IDs; when users choose to offload computation tasks, local resources are idle (Full offload, FO).

In Fig. 3, we show the system average cost and throughput of different scenarios under the same parameters. In these scenarios, we set the data size of computation tasks generated by IVs to 7800KB, and the number of CPU cycles required to execute the task to 18570 Megacycles. Moreover, the number of IVs in the system is $N = 100$, and the IDs that can provide computing services for these users include $M = 9$ UAVs and one MEC server. In Fig. 3, IV chooses to execute computation task locally, which results in a significant increase in system cost, and the throughput is the lowest compared with other scenarios. In addition, the proposed MITA algorithm shows excellent performance under the same parameters, which can reduce the system average cost and improve the system throughput to the greatest extent. The figures also show that NU outperforms the average cost and throughput of other scenarios except for MITA. Because NU also uses the task allocation method of MITA algorithm. However, as no UAV can provide services for users in NU, by comparing NU with MITA, it can be proved that the existence of UAV is beneficial to the system. Furthermore, the comparison of FO and MITA can prove that MITA algorithm is effective to minimize system cost and improve throughput. As shown in Fig. 3 (a) and (b), MITA considers offloading partial computation task to multiple IDs that can establish communication links and deploying UAVs to provide computing services for IVs, which can effectively increase the throughput and reduce the system costs. Results show that the average cost of the proposed MITA is 39.7%, 10.6%, and 13.4% lower than that of LC, NU, and FO, respectively, while the throughput of LC, NU and FO is 71.0%, 43.2% and 45.6% lower than MITA.

Fig. 4 illustrates the impact of computation tasks of different sizes generated by IVs on the system average cost and throughput. With the increase of the size of the computation task data generated by IVs, the average cost of the system presents an upward trend in and the throughput also decreases. The increases in data size also push up the performance of MITA. In terms of system average cost and throughput, MITA shows better performance compared with other scenarios. It can be concluded from Fig. 4 (a) that with the increase of data size, the cost of optimizing the system by MITA is more obvious than other strategies. In addition, Fig. 4 (b) shows that the MITA task allocation algorithm significantly improves throughput as the data size is small. It can be seen that MITA can greatly optimize the task execution efficiency under the same conditions. The numerical results show that, with the increase of task data size, the system average cost of our proposed MITA is decreased from 13.8% to 37.1% compared with other scenarios, while the throughput of other scenarios is decreased from 27.9% to 67.7% compared with MITA.

In order to evaluate the impact of the number of IVs in the system on the vehicular computation task offloading system, we also evaluate the system average cost and throughput with different number of IVs in Fig. 5 (a). With the increase in the number of IVs, the throughput of our proposed MITA task allocation policy has maintained an upward trend in throughput, and the system average cost has also shown an upward trend. This is because as the number of vehicles increases, the number of computation tasks to be processed grows, then the number of tasks in the waiting queue increases, thus increasing the cost of execution time. Moreover, in Fig. 5 (b), the influence of the number of UAVs on the vehicular computation task offloading system is investigated. According to the results, with the increase in the number of UAVs, the average cost of the system presents a gradual downward trend, while the throughput gradually increases. It can be concluded that although the increase of UAV number has enriched the computing resources of the system, it also increased the energy consumption of the system. Therefore, reasonable increment of the number of UAVs is the key to the effective implementation of multiple UAVs assisted vehicular computation offloading.

Next, we will discuss the performance comparison between our proposed MITA and the task allocation algorithms [9] and

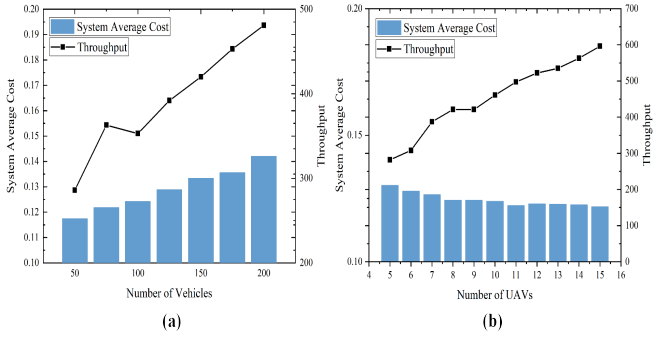

Fig. 5. System average cost and throughput vs. the number of IVs and UAVs

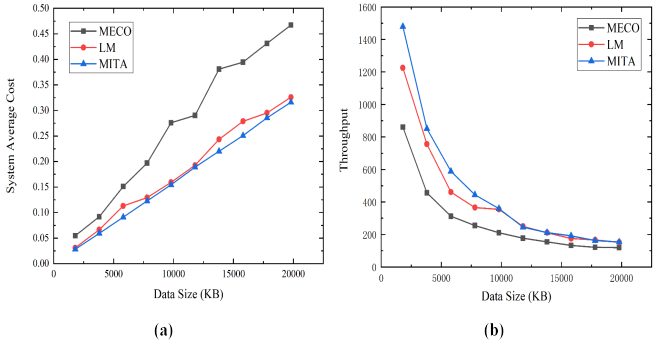

Fig. 6. The performance comparison of MITA and related work

[23]. The algorithm latency minimization (LM) [23] is to minimize the execution time under the premise of ensuring the maximum energy consumption, by considering offloading the computation tasks of mobile users to multiple cloud servers. In addition, the task allocation algorithm MECO [9] is to offload computation task of mobile user to an optimal intelligent device. The simulation results are shown in Fig.6 (a) and Fig.6 (b). The proposed MITA algorithm calculates the optimal task distribution method as the size of the computation task changes, and it shows good performance in terms of system average cost and throughput. As shown in Fig. 6, the proposed MITA performs better than MECO which proves distributing computation task to multiple IDs is a wise way. Moreover, the LM algorithm is based on the idea of offloading computation task to multiple cloud servers at the same time, so as to minimize the execution time of tasks and formulate a not bad allocation scheme. However, the idea of our proposed task allocation algorithm MITA is to allocate the task of unit size to the device with minimum cost increment. Hence, MITA can ensure that the task allocation scheme can make the system cost as small as possible and improve the throughput.

VII. Conclusion

This paper investigates the offloading strategy for industrial sensing vehicles in temporary manufacturing sites with less infrastructure by introducing multiple UAVs to assist the job. It is considered that IV can assign the computation task to multiple IDs that can establish wireless communication. Then, these available IDs and the vehicle itself can execute computation tasks in parallel, to maximize the utilization of system computing resources. This can improve the computing efficiency and minimize the system average cost. Based on the idea of sequence game in the SDN framework, we make the decision of players' mutual satisfaction and propose the MITA algorithm to allocate computation tasks to multiple available devices to minimize the system cost. As one of the cost elements, we add the computing services provided by hiring IDs, which is closer to the reality that the temporal industrial site can contain multiple manufacturers. Numerical results show that deploying multiple UAVs to provide computing services is a reliable and low-cost solution to satisfy the IV with scared computational resources. Moreover, the proposed MITA algorithm can greatly minimize the average cost of computation tasks generated by IVs and improve system throughput.

References

[1] Z. Ning, X. Hu, Z. Chen, M. Zhou, B. Hu, J. Cheng, and M. S. Obaidat, "A cooperative quality-aware service access system for social internet of vehicles," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2506–2517, 2018.

[2] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[3] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, pp. 89–103, 2015.

[4] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 5529–5534.

[5] H. Li, M. Dong, and K. Ota, "Control plane optimization in software-defined vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7895–7904, 2016.

[6] Q. Zheng, K. Zheng, H. Zhang, and V. C. M. Leung, "Delay-optimal virtualized radio resource scheduling in software-defined vehicular networks via stochastic learning," *IEEE Transactions on Vehicular Technology*, pp. 7857–7867, 2016.

[7] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, pp. 84–91, 2016.

[8] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Vehicular Technology Magazine*, pp. 73–82, 2017.

[9] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[10] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2019.

[11] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1202–1207.

[12] A. M. Vegni and V. Loscrí, "A survey on vehicular social networks," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2397–2419, 2015.

[13] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, 2016.

[14] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[15] R. Duan, J. Wang, J. Du, C. Jiang, T. Bai, and Y. Ren, "Power-delay trade-off for heterogenous cloud enabled multi-uav systems," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[16] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[17] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[18] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[19] X. Yang, Z. Li, and X. Ge, "Deployment optimization of multiple uavs in multi-uav assisted cellular networks," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1–7.

[20] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-uav enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[21] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, 2018.

[22] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in uav-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147–3159, 2020.

[23] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[24] S. Kim, S. Park, M. Chen, and C. Youn, "An optimal pricing scheme for the energy-efficient mobile edge computation offloading with ofdma," *IEEE Communications Letters*, vol. 22, no. 9, pp. 1922–1925, 2018.

[25] L. Zhang, Z. Zhao, Q. Wu, H. Zhao, H. Xu, and X. Wu, "Energy-aware dynamic resource allocation in uav assisted mobile edge computing over social internet of vehicles," *IEEE Access*, vol. 6, pp. 56 700–56 715, 2018.

[26] X. Kang, R. Zhang, and M. Motani, "Price-based resource allocation for spectrum-sharing femtocell networks: A stackelberg game approach," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 538–549, 2012.

[27] Y. Wu, T. Zhang, and D. H. K. Tsang, "Joint pricing and power allocation for dynamic spectrum access networks with stackelberg game model," *IEEE Transactions on Wireless Communications*, vol. 10, no. 1, pp. 12–19, 2011.

[28] X. Wang and L. Duan, "Dynamic pricing and capacity allocation of uav-provided mobile services," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1855–1863.

[29] A. Alioua, S. Senouci, S. Moussaoui, H. Sedjelmaci, and M. Messous, "Efficient data processing in software-defined uav-assisted vehicular networks: A sequential game approach," *Wireless Personal Communications*, vol. 101, pp. 2255–2286, 2018.