sensors

*Article*

# Computation Offloading in UAV-Enabled Edge Computing: A Stackelberg Game Approach

**Xinwang Yuan, Zhidong Xie * and Xin Tan**

National Innovation Institute of Defense Technology, Academy of Military Science of the People's Liberation Army, Beijing 100071, China; alter_yxw@163.com (X.Y.); tanxin2017@163.com (X.T.)
* Correspondence: xzd313@163.com

**Abstract:** This paper studies an efficient computing resource offloading mechanism for UAV-enabled edge computing. According to the interests of three different roles: base station, UAV, and user, we comprehensively consider the factors such as time delay, operation, and transmission energy consumption in a multi-layer game to improve the overall system performance. Firstly, we construct a Stackelberg multi-layer game model to get the appropriate resource pricing and computing offload allocation strategies through iterations. Base stations and UAVs are the leaders, and users are the followers. Then, we analyze the equilibrium states of the Stackelberg game and prove that the equilibrium state of the game exists and is unique. Finally, the algorithm's feasibility is verified by simulation, and compared with the benchmark strategy, the Stackelberg game algorithm (SGA) has certain superiority and robustness.

**Keywords:** mobile edge computing; unmanned aerial vehicles; computation offloading; Stackelberg game

check for updates

**Citation:** Yuan, X.; Xie, Z.; Tan, X. Computation Offloading in UAV-Enabled Edge Computing: A Stackelberg Game Approach. *Sensors* **2022**, *22*, 3854. https://doi.org/10.3390/s22103854

Academic Editors: Diego González-Aguilera, Pablo Rodríguez-Gonzálvez and George Nikolakopoulos

Received: 30 March 2022
Accepted: 17 May 2022
Published: 19 May 2022

## 1. Introduction

With the rapid development of network communication technology, the data interaction efficiency of the mobile Internet is constantly improving. Meanwhile, the transmission bandwidth and data scale have become increasingly large. 5G communication and cloud computing have spawned new applications such as driverless, automatic navigation, face recognition, and augmented reality [1]. Meanwhile, these applications are computationally intensive and time-sensitive. However, mobile devices at the terminal cannot provide sufficiently high-performance computing services, and the battery capacity is limited, so it is inefficient in handling these tasks and may not meet the quality of services requirements [2]. In the network architecture of cloud computing, computing resources concentrate in the cloud, and there is a certain distance between the computing resources and terminal devices. Therefore, the service response has an inevitable delay [3], and when dealing with computationally intensive tasks; it is prone to access congestion.

Mobile edge computing (MEC) is a new computing architecture for providing computing services [4,5] that can push the service resources of cloud computing to the edge to meet the requirements of intensive computing and low latency. Compared with cloud computing, edge computing is more in line with the concept of a smart city, which is proposed to realize green and sustainable development [6]. Traditional terrestrial networks face challenges in scenarios such as complex terrain and equipment failure. Unmanned aerial vehicles (UAV) can help to enhance the flexibility and robustness of mobile edge computing network deployment [7], and reduce the complexity and cost of resource management. For example, Verizon and AWS work together to combine UAVs with mobile edge computing to reduce connection latency and reduce UAV costs by about 10% [8]. However, with the increasing number of UAVs in use, resource management of networks faces challenges such as power control, spectrum allocation, interference management, and task allocation [9].

In terms of task allocation, different devices in the MEC network differ in various aspects, such as onboard battery capacity and computational performance. Tasks that require high performance can be offloaded to devices with large battery capacity and high computing performance. In MEC networks, the offloading forms of computation include partial offloading (complex computation tasks can be divided into several sub-computation tasks.) and full offloading (tasks are inseparable) [10]. The computation task offload strategy should consider determining the proportion of the offloading task and choosing the offloading computation content [11]. Moreover, the balance between throughput and fairness also needs to be considered [12]. Artificial intelligence (AI) could also help to allocate resources dynamically. Combined with AI, the learning capacity of edge devices could be enhanced [13]. As shown in Figure 1, the UAV in the MEC network can act as a user at the terminal to access the MEC service or a server to receive the tasks from users; it can also act as a relay node to forward tasks offloaded from nearby users to the base station servers. Roles are changed according to the scenarios.

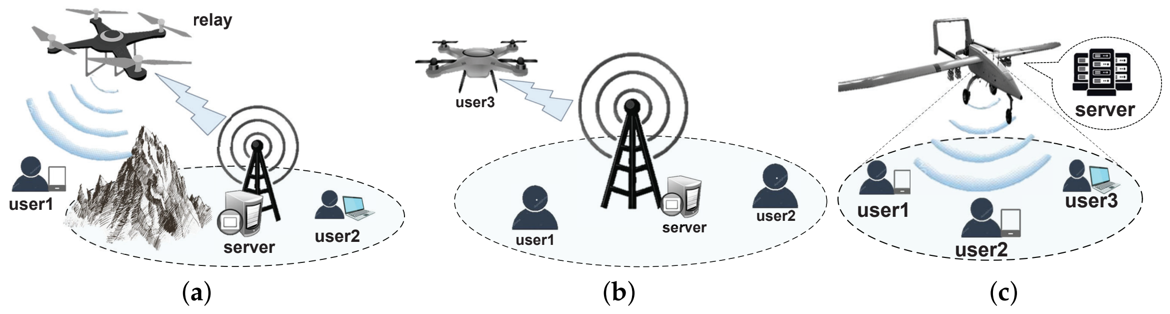

**Figure 1.** The roles of UAVs in computing resource allocation for MEC networks. (**a**) uav as relay. (**b**) uav as user. (**c**) uav as server.

## *1.1. Related Works*

To improve the performance of the computational tasks' allocation strategy, it is necessary to set the optimization goal first. On the one hand, some studies considered a single performance metric. Long et al. [14] aimed to minimize the computation latency and obtain the optimal offloading computation task strategy. However, it is necessary to consider introducing a UAV-enabled MEC network to improve the performance of computing offload further. Luan et al. [15] aimed to minimize the energy consumption and solve the problem of distributed task allocation of UAVs based on a coalitional game. On the other hand, some studies have integrated multiple performance metrics, which could improve the system computing performance more comprehensively. Chen et al. [16] integrated computing latency and energy consumption, and then introduced the pricing of computing resources to differentiate the servers on base stations and UAVs, but it is also necessary to analyze the allocation scheme of the optimal offloading proportion. Ren et al. [17] considered to minimize the global computation latency and energy consumption and divided the computation task into several subtasks, then worked out the optimal offloading proportion by KKT-condition. In addition, in that study, the optimal offloading strategy and minimum computing latency were obtained by a non-cooperative game. However, due to the limited endurance of UAVs, a single UAV cannot provide services for ground users efficiently, so the performance of multiple UAVs should also be considered.

In addition, some studies have considered the resource allocation in the more complex network environment. Alioua. et al. [18] comprehensively considered latency, energy consumption, link quality, and other factors in a multi-UAV-enabled road traffic monitoring scenario, and obtained a computation resource allocation strategy with better overall performance based on the sequential game. EI N.N et al. [19] considered device association, task assignment, and computational resource allocation problems comprehensively, and proposed an iterative algorithm based on block coordinate descent to minimize the energy consumption of mobile devices and UAVs. Yan et al. [20] considered the MEC network of a multi-UAV and multi-ground user scenario and constructed a Stackelberg game model: The UAV, as the vice-leader, maximizes the income by optimizing the location and pricing of computing resources. In addition, the user, as the follower, reduces the overhead by optimizing the offload allocation strategy. Dong et al. [21] points out that most studies focus on simple application scenarios, and the research on multi-UAV cooperative resource management in complex environments needs to be further deepened and improved.

### *1.2. Contributions*

According to related works above and [22–26], game theory has a good application prospect in solving resource management of the MEC network recently. In addition, we found that multiple UAV applications should be considered in MEC resource management, and energy consumption, delay, and other aspects should be considered comprehensively in performance indexes to improve system performance comprehensively. In Stackelberg game, the player can decide the strategy at different layers. The followers respond to the leader's strategic actions and choose the best strategy. Therefore, players at all levels can consider their own interests and set different utility functions. Previous studies [27,28] have used the Stackelberg game to solve the problem of computing unloading in MEC, we extend it to the UAV-enabled application scenario, and it is different that we divide players into three layers based on the supply of computation. The contributions of this study are as follows:

1. In the multi-UAV cooperative MEC network, we construct a multi-layer Stackelberg game model according to the different characteristics of base stations, UAVs, and users. And the equilibrium state is reached through several iterations to meet the maximum interests of players in each layer.
2. Players at each layer of the Stackelberg game set the utility function according to their own interests and demands, and select the optimal computational offloading strategy to maximize the total utility of the system.
3. Performance metrics such as computation latency and energy consumption are weighted to improve the performance of the system more comprehensively.
4. The proposed SGA has a certain superiority and robustness. Compared with the benchmark scheme of random pricing strategy, it can achieve higher total system benefits in multiple different scenarios.

## 2. Model of Stackelberg Game

The main symbols in the model are given in Nomenclature.

### *2.1. Leader Sub-Game*

In the MEC network, the ground base station has strong computing ability and continuous power supply, so the base stations act as the leader and the UAV as the vice-leader in the Stackelberg game. For each leader station $g$, the onboard edge server can receive computing tasks directly from the user or employ the UAVs as relay nodes to forward computing tasks from the users. The leader needs to set a different price $M_i^g$ for each computing resource requested by user $i$, a corresponding price $M_j^g$ for the employed relay UAV $j$. The computing resource $F_g$ of the leader is limited, and the computing resource allocated to the user $i$ is $F_i^g$. Then we can obtain the profit $P_g$ that the base station by providing computing services to the user and the cost $C_g$ of employing the UAV as follows:

$$P_g = \sum_{i \in \mathcal{I}} M_i^g F_i^g Need_i^g \tag{1}$$

$$C_g = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} M_j^g D_i Need_i^j \gamma_i^r \tag{2}$$

where $D_i$ represents the computation task quantity of user $i$, $Need_i^g = 1$ indicates that user $i$ needs to offload the computing task to base station g, $Need_i^j = 0$ indicates that the computing task of user $i$ needs the assistance of UAV $j$, and $\gamma_i^r$ indicates the proportion that user $i$ chooses to forward the computing task to the base station through relay.

Then, the net income of the leader base station $g$ can be expressed as $U_g = P_g - C_g$, and the game of the leader layer can be formulated as:

$$\max U_g(\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{F_i^g}), \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{3}$$

$$s.t. \begin{cases} \sum_{i \in \mathcal{I}} F_i^g \leq F_g & F_i^g \leq D_i \\ M_i^g \in [\min M_i^g, \max M_i^g] & \\ M_j^g \in [\min M_j^g, \max M_j^g] & \\ j_g \bigcap j_{g'} = \varnothing & \forall g, g' \in \mathcal{G} \end{cases} \tag{4}$$

where $\boldsymbol{M_i^g}$ is the pricing set of all base stations for user $i$, $\boldsymbol{M_j^g}$ is the base station's pricing set for hiring UAVs, $\boldsymbol{F_i^g}$ is the resource set allocated to the user by the base station, $\mathcal{I}, \mathcal{J}, \mathcal{G}$ are respectively the set of users, UAVs, and base stations. The first constraint indicates that the computing resources of the base station are limited, the second and third constraints indicate that the pricing for user services and hiring UAVs should fall within a certain range, and the fourth constraint indicates that each UAV can only be employed by one base station.

## *2.2. Vice-Leader Sub-Game*

In the MEC network, the UAV can act as a server to receive computing service requests from users or as a relay node to forward service requests from users to the base station. $Role_i^j = 1$ represents that the UAV $j$ acts as a server, and $Role_i^j = 0$ represents a relay node when processing the computing tasks of user $i$. The benefits of these two roles are denoted as $P_j^{compute}$ and $P_j^{relay}$, respectively. When acting as a server, the UAV sets a price $m_i^j$ for the computing resources accessed by different users. As with the BS in the leader layer, the service resource $F_j$ of the UAV is also limited, and the resource allocated to user $i$ is denoted as $f_i^j$. Then the profit $P_j^{compute}$ and $P_j^{relay}$ of the vice-leader can be expressed as:

$$P_j^{compute} = \sum_{i \in \mathcal{I}} m_i^j f_i^j Role_i^j \tag{5}$$

$$P_j^{relay} = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} M_j^g D_i Need_i^j \left(1 - Role_i^j\right) \tag{6}$$

When UAV $j$ acts as an edge server, there is overhead $C_j^{compute}$ in processing computation tasks, and when acting as a relay, there is the communication cost $C_j^{trans}$ in transmitting the tasks:

$$C_j^{compute} = \sum_{i \in \mathcal{I}} \frac{\varphi_i}{f_i^j} \eta_i Need_i^j Role_i^j \tag{7}$$

$$C_j^{trans} = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} p_j \frac{D_i}{rate_i^{j,g}} Need_i^j \left(1 - Role_i^j\right) \tag{8}$$

Therefore, the profit of the vice-leader UAV $j$ can be expressed as $U_j = P_j^{compute} + P_j^{relay} - C_j^{compote} - C_j^{trans}$, and the game of the vice-leader can be formulated as:

$$\max U_j(\boldsymbol{m_i^j}, \boldsymbol{f_i^j}), \forall i \in \mathcal{I} \tag{9}$$

$$s.t.\begin{cases} Role_i^j \in \{0;1\} \\ m_i^j \in [\min m_i^j, \max m_i^j] \\ \sum_{i\in\mathcal{I}} f_i^j \le F_j, f_i^j \le D_i \end{cases} \tag{10}$$

### 2.3. Follower Sub-Game

As a follower, user $i$ taking the fees $C_i^{pay}$ paid for computation services and the processing cost $C_i^{compute}$ (including the computation latency and communication cost in transmission) into account and decides the offloading object and proportion based on the pricing strategy given in the leader layer and the allocated computing resources.

$$C_i^{pay} = \begin{cases} 0, & \gamma_i^l \ge 0 \\ \alpha_{g,i}^{pay} M_i^g F_i^g, & \gamma_i^g \ge 0 \\ \alpha_{j,i}^{pay} m_i^j f_i^j, & \gamma_i^j \ge 0 \\ \alpha_{relay,i}^{pay} M_i^g F_i^g, & \gamma_i^r \ge 0 \end{cases} \tag{11}$$

$$C_i^{compute} = \begin{cases} \alpha_{local,i}^{trans} \frac{\varphi_i}{f_i} + \alpha_{local,i}^{exe} \frac{\varphi_i}{f_i} \eta_i, & \gamma_i^l \ge 0 \\ \alpha_{g,i}^{time} \left( \frac{D_i}{Rate_i^g} + \frac{\varphi_i}{F_i^g} \right) + \alpha_{g,i}^{energy} p_i \frac{D_i}{F_i^g}, & \gamma_i^g \ge 0 \\ \alpha_{j,i}^{time} \left( \frac{D_i}{Rate_i^j} + \frac{\varphi_i}{f_i^j} \right) + \alpha_{j,i}^{energy} p_i \frac{D_i}{f_i^j}, & \gamma_i^j \ge 0 \\ \alpha_{relay,i}^{time} \left( \frac{D_i}{Rate_i^j} + \frac{\varphi_i}{F_i^j} + \frac{D_i}{Rate_i^{j,g}} \right) + \alpha_{relay,i}^{energy} p_i \frac{D_i}{Rate_i^j}, & \gamma_i^r \ge 0 \end{cases} \tag{12}$$

where $p_i$ denotes the user's transmission power, and $\gamma_i^l$, $\gamma_i^g$, $\gamma_i^j$ and $\gamma_i^r$ denote the allocation ratios for local computing, offloading to the base station, offloading to the UAV and forwarding to the base station via relay, respectively. Then, the utility function of user $i$ can be expressed as: $U_i = -C_i^{compute} - C_i^{pay}$, and the game of the follower can be defined as:

$$\max U_i \left( \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r \right) \tag{13}$$

$$s.t.\begin{cases} \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r \in [0,1] \\ \gamma_i^l + \gamma_i^g + \gamma_i^j + \gamma_i^r = 1 \end{cases} \tag{14}$$

### 2.4. Computation Offload Model Based on Stackelberg Game

The game of base station, UAV, and user from the first three sections can form a three-layer heterogeneous Stackelberg game model, which can be expressed as:

$$\boldsymbol{\mathcal{G}} = \left\{ (\mathcal{G}, \mathcal{J}, \mathcal{I}), (\mathbb{G}, \mathbb{J}, \mathbb{I}), (U_g, U_j, U_i) \right\} \tag{15}$$

where $\mathbb{G}, \mathbb{J}, \mathbb{I}$ are the sets of strategy space, respectively. For this Stackelberg game model, to get the state of Stackelberg Equilibrium(SE), the set $\left( \boldsymbol{M_i^{g*}}, \boldsymbol{M_j^{g*}}, \boldsymbol{m_i^{j*}}, \boldsymbol{F_i^{g*}}, \boldsymbol{f_i^{j*}}, \gamma_i^{l*}, \gamma_i^{g*}, \gamma_i^{j*}, \gamma_i^{r*} \right)$ should satisfied the conditions as follow:

$$U_g \left( \boldsymbol{M_i^{g*}}, \boldsymbol{M_j^{g*}}, \boldsymbol{F_i^{g*}}, \mathbb{J}^*, \mathbb{I}^* \right) \ge U_g \left( \boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{F_i^g}, \mathbb{J}^*, \mathbb{I}^* \right) \tag{16}$$

$$U_j \left( \mathbb{G}^*, \boldsymbol{m_i^{j*}}, \boldsymbol{f_i^{j*}}, \mathbb{I}^* \right) \ge U_j \left( \mathbb{G}^*, \boldsymbol{m_i^j}, \boldsymbol{f_i^j}, \mathbb{I}^* \right) \tag{17}$$

$$U_i \left( \mathbb{G}^*, \mathbb{J}^*, \gamma_i^{l*}, \gamma_i^{g*}, \gamma_i^{j*}, \gamma_i^{r*} \right) \ge U_i \left( \mathbb{G}^*, \mathbb{J}^*, \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r \right) \tag{18}$$

## 3. Equilibrium Analysis of Stackelberg Game

### *3.1. Existence Analysis of the Equilibrium Solution*

**Theorem 1.** *Stackelberg Equilibrium point exists in the Stackelberg multi-layer game $\boldsymbol{\mathcal{G}}$.*

**Proof.** As the leader, the utility of base station $\mathcal{G}$ is:

$$\begin{aligned} U_g\left(\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{F_i^g}, \mathbb{J}^*, \mathbb{I}^*\right) &= P_g - C_g \\ &= \sum_{i\in\mathcal{I}} M_i^g F_i^g Need_i^g - \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} M_j^g D_i Need_i^j \gamma_i^r \end{aligned} \tag{19}$$

get the partial derivatives respectively, there are:

$$\begin{aligned} \frac{\partial U_g}{\partial \boldsymbol{M_i^g}} &= \boldsymbol{F_i^g} Need_i^g \geq 0 \\ \frac{\partial U_g}{\partial \boldsymbol{F_i^g}} &= \boldsymbol{M_i^g} Need_i^g \geq 0 \\ \frac{\partial U_g}{\partial \boldsymbol{M_j^g}} &= -D_i Need_i^j \gamma_i^r \leq 0 \end{aligned} \tag{20}$$

therefore, $U_g$ is monotonous to $\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{F_i^g}$, then $\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{F_i^g}$ are bounded could be deduced according to the constraint condition (4). The condition (16) is satisfied when $\boldsymbol{M_i^{g*}} = \max M_i^g, \boldsymbol{M_j^{g*}} = \min M_j^g$ and $\boldsymbol{F_i^{g*}} = \max F_i^g$.

As the vice-leader, the utility of UAV $\mathcal{J}$ is:

$$\begin{aligned} U_j\left(\mathbb{G}^*, \boldsymbol{m_i^j}, \boldsymbol{f_i^j}, \mathbb{I}^*\right) &= P_j^{compute} + P_j^{relay} - C_j^{compute} - C_j^{trans} \\ &= \sum_{i\in\mathcal{I}} m_i^j f_i^j Role_i^j + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} M_j^g D_i Need_i^j\left(1 - Role_i^j\right) \\ &- \sum_{i\in\mathcal{I}} \frac{\varphi_i}{f_i^j} \eta_i Need_i^j Role_i^j - \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_j \frac{D_i}{Rate_i^j} Need_i^j\left(1 - Role_i^j\right) \end{aligned} \tag{21}$$

get the partial derivatives respectively, there are:

$$\begin{aligned} \frac{\partial U_j}{\partial \boldsymbol{m_i^j}} &= \boldsymbol{f_i^j} Role_i^j \geq 0 \\ \frac{\partial U_j}{\partial \boldsymbol{f_i^j}} &= m_i^j Role_i^j + \frac{\varphi_i}{\left(f_i^j\right)^2} \eta_i Need_i^j Role_i^j \geq 0 \end{aligned} \tag{22}$$

therefore, $U_j$ is monotonous to $\boldsymbol{m_i^j}, \boldsymbol{f_i^j}$, then $\boldsymbol{m_i^j}, \boldsymbol{f_i^j}$ are bounded could be deduced according to the constraint condition (10). Let $\boldsymbol{m_i^j} = \max m_i^j, \boldsymbol{f_i^j} = \max f_i^j$, the condition (17) is satisfied.

As follower, the utility of user $\mathcal{I}$ is:

$$U_i\left(\mathbb{G}^*, \mathbb{J}^*, \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r\right) = \left[\begin{array}{cccc} \gamma_i^l & \gamma_i^g & \gamma_i^u & \gamma_i^r \end{array}\right] \cdot \left(-C_i^{compute} - C_i^{pay}\right) \tag{23}$$

since $C_i^{compute}$ and $C_i^{pay}$ are constant for $\boldsymbol{\gamma_i}$, $U_i$ could be simplified as:

$$U_i\left(\mathbb{G}^*, \mathbb{J}^*, \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r\right) = C_1\gamma_i^l + C_2\gamma_i^g + C_3\gamma_i^j + C_4\gamma_i^r \tag{24}$$

therefore, $U_i$ is linearly related to $\gamma_i$, and the game of this layer is a linear programming problem. The constraint given by (14) is a convex set, then the objective function $U_i$ obtains the optimal value at the vertex of the feasible domain [29].

In summary, the Stackelberg Equilibrium point exists in multi-layer game $\mathcal{G}$. □

### 3.2. Equilibrium Uniqueness Analysis of Stackelberg Game

**Theorem 2.** *The stackelberg game equilibrium point is unique.*

**Proof.** For the leader and vice-leader layer, it could be known from (20) and (22) that $U_g$ and $U_j$ are monotonic to $\boldsymbol{M_i^g}, \boldsymbol{M_i^j}, \boldsymbol{F_i^g}$ and $\boldsymbol{m_i^j}, \boldsymbol{f_i^j}$. Therefore, the optimal solutions obtained are unique.

For follower layer, according to the constraint (14), the feasible domain of $\gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r$ are convex sets, and the optimal solution is obtained at the vertex. In addition, the feasible domain formed by constraint (14) is a pyramid, so the vertex is unique, and the optimal solution of the follower layer is unique [29].

In summary, the Stackelberg Equilibrium point in multi-layer game $\mathcal{G}$ is unique. □

## 4. Stackelberg Game Resource Pricing and Computation Allocation Algorithm

1. Parameters initialization, initialize the values of the parameters in the simulation environment and the strategy for the first round of the game. Get the distances between each base station, UAV and user. And users will choose to establish a connection with the nearest base station or UAV in the initial state.
2. Resource pricing and allocation, the players in each layer play a game of resource allocation and pricing, updating the optimal strategy based on the current round in the order of leader, vice-leader, and follower, $\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{m_i^j}, \boldsymbol{F_i^g}, \boldsymbol{f_i^j}, \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r$ are involved (as in Algorithm 1).
3. Repeat step (2) for iterations, and eventually converge to equilibrium through multiple rounds of the game.

**Algorithm 1** Stackelberg game resource pricing and allocation algorithm

```
    Input: Location: $bs, uav, user$; Computation amount: $D_i$; Iteration number: epoch
    Output: Price: $\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{m_i^j}$; Resource amount: $\boldsymbol{F_i^g}, \boldsymbol{f_i^j}$; Offload rate:$\gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r$
1: Simulation parameter initialized according to the distance relationships and computation amount.
2: for episode = 1 to epoch do
3:     for b = 1 to $bs_{num}$ do
4:         $F_i^g = D_i \cdot \left(\gamma_i^g + \gamma_i^r\right)$;
5:         if $\gamma_i^g \geq 0$ and $Need_i^g$==1 then
6:             if min $M_i^g \leq 2M_i^g \leq$ max $M_i^g$ then
7:                 $M_i^g$=2 $\cdot$ $M_i^g$; $M_{i,low}^g$=$M_i^g$;
8:         else
9:             $M_{i,high}^g$=$M_i^g$; $M_i^g$=$\frac{M_{i,high}^g + M_{i,low}^g}{2}$;
10:        if $\gamma_i^r \geq 0$ and $Need_i^g$ == 1 then
11:            $M_{j,high}^g$=$M_j^g$; $M_j^g$=$\frac{M_{j,high}^g + M_{j,low}^g}{2}$
12:        else
13:            if min $M_j^g \leq 2M_j^g \leq$ max $M_j^g$ then
14:                $M_{j,low}^g$=$M_j^g$; $M_j^g$=2 $\cdot$ $M_j^g$
15:    for u = 1 to $uav_{num}$ do
16:        $f_i^j$=$D_i \cdot \gamma_i^j$
17:        if $\gamma_i^j \geq 0$ and $Need_i^j$==1 then
```

```
18:             if $\min m_i^g \le 2m_i^g \le \max m_i^g$ then
19:                 $m_i^g = 2 \cdot m_i^g$; $m_{i,low}^g = m_i^g$;
20:         else
21:             $m_{i,high}^g = m_i^g$; $m_i^g = \frac{m_{i,high}^g + m_{i,low}^g}{2}$;
22:     for i = 1 to $user_{num}$ do
23:         update $\gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r$ by utility function according to $\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{m_i^j}, \boldsymbol{F_i^g}, \boldsymbol{f_i^j}$;
24:         update $Need_i^g, Need_i^j$ according to $\gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r$.
25:     record the utility of each player in this iteration.
26: return $\boldsymbol{M_i^g}, \boldsymbol{M_j^g}, \boldsymbol{m_i^j}, \boldsymbol{F_i^g}, \boldsymbol{f_i^j}, \gamma_i^l, \gamma_i^g, \gamma_i^j, \gamma_i^r$ and the final utility of each players.
```

## 5. Results

### 5.1. Simulation Parameter Setting

The basic simulation environment is a MEC network composed of two base stations, 6 UAVs, and eight users. The locations of the base stations, UAVs, and users (randomly generated) are distributed in a Cartesian plane coordinate system as shown in Figure 2. The computing resources of base stations, UAVs, and users are all 64 GB, 8 GB, and 32 MB, respectively. In addition, the computation amount of users are random in 10 MB∼1 GB. Other parameters are shown in Table 1. The program design and result diagram of the simulation are completed in MATLAB, and the operating system is Windows 10, 64-bit professional version. The base stations, UAVs, and users in the simulation scene are virtual devices imitating real device parameters.

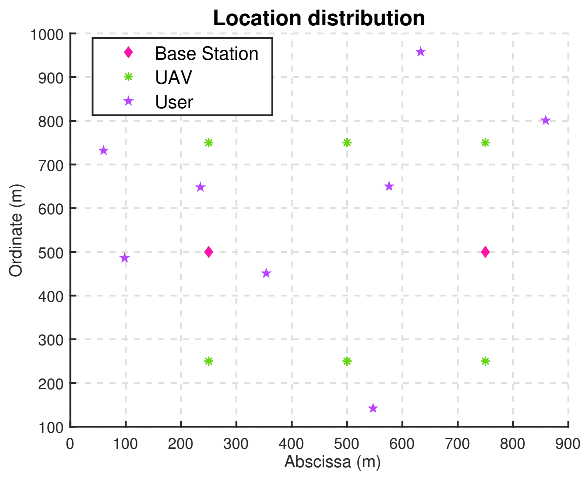

**Figure 2.** Location distribution in simulation (UAVs' coordinates are taken from ground projection).

**Table 1.** Other simulation parameter settings.

| Parameter | Symbolic Representation | Value |
|---|---|---|
| Energy consumption per CPU cycle | $\eta_i$ | 8 |
| Weights about local computation | $\alpha_{local,i}^{time}, \alpha_{local,i}^{energy}$ | (0.5, 0.5) |
| Weights about offload to BS | $\alpha_{g,i}^{time}, \alpha_{g,i}^{energy}, \alpha_{g,i}^{pay}$ | (0.3, 0.3, 0.4) |
| Weights about offload to UAV | $\alpha_{j,i}^{time}, \alpha_{j,i}^{energy}, \alpha_{j,i}^{pay}$ | (0.3, 0.3, 0.4) |
| Weights about relay | $\alpha_{relay,i}^{time}, \alpha_{relay,i}^{energy}, \alpha_{relay,i}^{pay}$ | (0.3, 0.3, 0.4) |

### 5.2. Simulation Result and Performance Comparison

Figure 3 shows the iteration of the three roles of the base station, UAV, and user in the Stackelberg multi-layer game. It can be found that after about 250 iterations, the income or overheads of players at each layer finally reached the equilibrium state, indicating that the algorithm can achieve Stackelberg equilibrium. During the whole process, as shown in Figure 3a, the incomes of BS1 and BS2 were basically stable, which changed suddenly at about 150 iterations and returned to equilibrium after about 50 iterations. In Figure 3b, the overhead of user2 changed dramatically about 150 iterations, indicating that user2 switched the offloading object. And after about 50 iterations, user2 reached the equilibrium state. The overheads of other users gradually increase in the oscillation and enter the equilibrium state when they reach the certain values, which indicates that the prices for computation services had reached the maximum value, and the offloading objects had not changed during that period. In Figure 3c, the incomes of UAVs fluctuated greatly before convergence, indicating that the competition of UAVs in the game of resource allocation was fierce. We can also find that after about 150 iterations, the incomes of UAVs were oscillations in several points, but as they gradually entered the equilibrium state, there were fewer and fewer oscillations until they reach the equilibrium and no longer oscillate. The specific computation offloading situation in Stackelberg equilibrium is shown in Figure 4.

The benchmark algorithm is RANDOM (the users decide the offload objects and proportions randomly) and LOCAL (all users execute the task locally), and the total profit of a multi-level game is used as the performance evaluation standard. The total profit of the system can be expressed as:

$$Utility = U_g + U_j - U_i \tag{25}$$

As shown in Figure 5, we compare the total system profits of the Stackelberg game algorithm (SGA) with RANDOM and LOCAL. Firstly, it can be found that the SGA strategy reached an equilibrium state after about 250 iterations. Secondly, the total system profit of SGA in the equilibrium state is about 80% more than that of random strategy, and the profits of SGA and RANDOM are more than LOCAL.

We also compare the SGA in an equilibrium state with the RANDOM and LOCAL strategy in five different scenarios, as shown in Figure 6. Figure 6a shows the total system profit in the scenarios where the quantity of users is 2, 4, 8, 12, and 16, respectively. We can see SGA is significantly better than RANDOM and LOCAL in all five scenarios, and the total profit of the system increases with the quantity of users. Figure 6b is the performance comparison chart of SGA, LOCAL, and RANDOM in scenarios with different user positions, and the coordinates of users in the other four scenarios are also randomly generated. It can be found that the SGA algorithm is superior to RANDOM and LOCAL in these scenarios. In addition, we can find that the total system profit of RANDOM fluctuates greatly, and the profits in some scenarios are not as good as the LOCAL. While the total system profit of SGA is more stable. Figure 6c is the performance comparison chart of 5 scenarios with different quantities of user computation. They are all randomly generated, with the value of (673,978,768,843,408,616,543,424), (971,609,720,303,460,49,386,362), (288,817,451,807,791,283,169,255), (638,425,906,418,255,540,938,661), (395,259,848,946,377,468, 482,576) MB, respectively. It can be found that SGA is better than RANDOM and LOCAL. Since the amount of computation directly affects the overhead, it is normal for the total system profit to fluctuate.

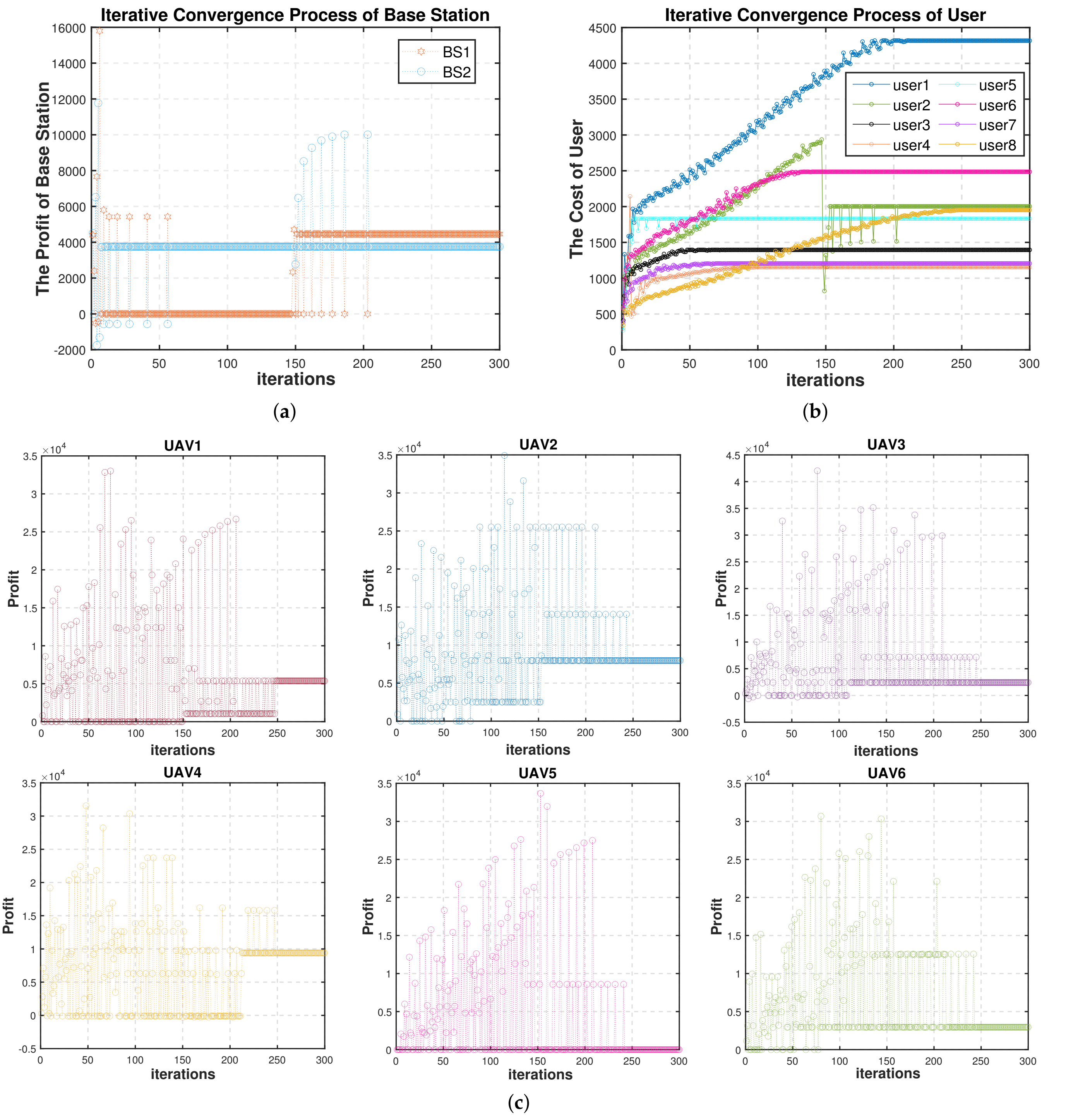

**Figure 3.** Iteration process in each layer. (**a**) iteration process of base stations. (**b**) iteration process of users. (**c**) iteration process of UAVs.

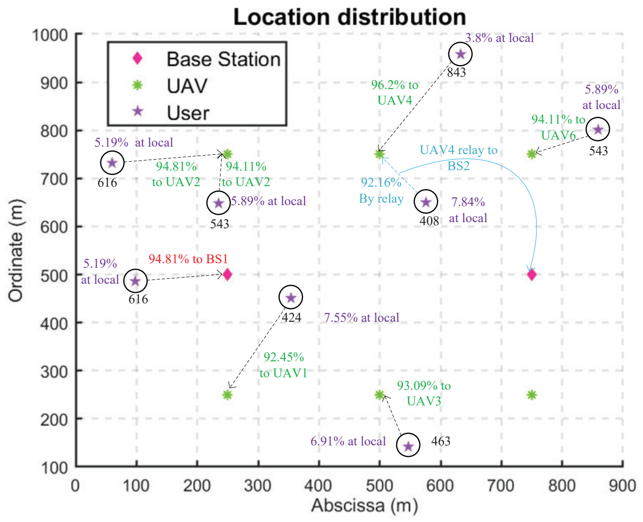

**Figure 4.** Optimal computation offload strategy.

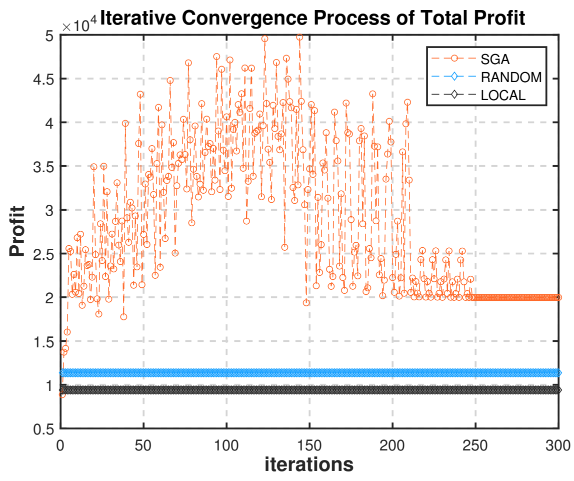

**Figure 5.** Iterative process of total profit of the system and comparison with random strategy.

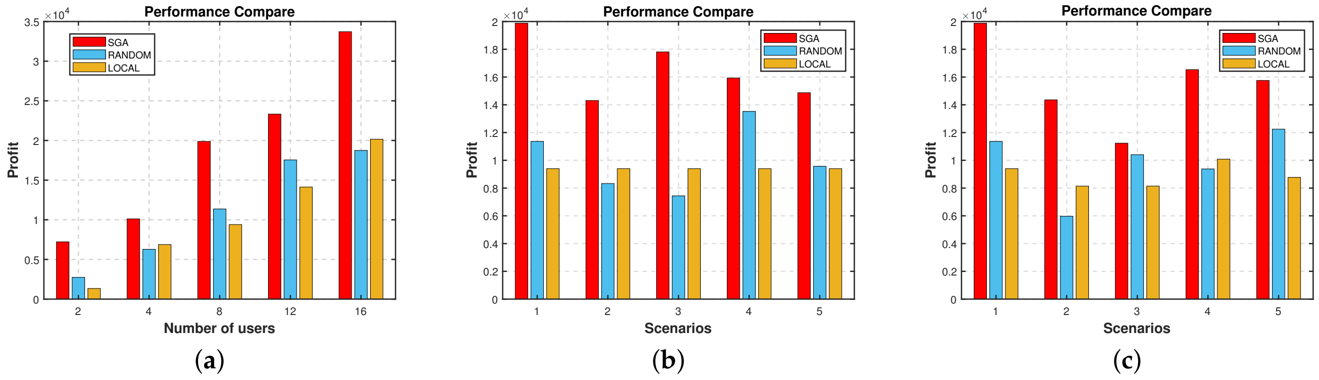

**Figure 6.** Performance comparison in different scenarios. (**a**) Different number of users. (**b**) Different user locations. (**c**) Different amount of computation.

## 6. Conclusions

This study aims to solve the problem of computing resource management in a multi-UAV-enabled edge computing scenario. According to the different interests of base stations, UAVs and users, we construct a three-layer Stackelberg game model by comprehensively considering computing latency, energy consumption, and specific profit. The equilibrium state is achieved after several iterations and verified by simulation. The Stackelberg game algorithm has better total system profit in multiple different scenarios and has certain robustness in the equilibrium state than the random strategy. Therefore, in the future scenario of multi-UAV-enabled edge computing, Stackelberg game theory has application prospects in solving complex resource problems.

**Author Contributions:** Conceptualization, X.Y.; methodology, X.Y.; software, X.Y.; validation, Z.X.; formal analysis, X.Y., X.T.; investigation, X.Y.; resources, Z.X.; data curation, X.Y., X.T.; writing—original draft preparation, X.Y.; writing—review and editing, Z.X.; visualization, X.Y.; supervision, Z.X.; project administration, Z.X.; funding acquisition, Z.X. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Nomenclature

| Parameter | Explanation |
|---|---|
| $\varphi_i$ | Required CPU cycles to complete the computation of user $i$ |
| $\eta_i$ | Energy consumption per unit CPU cycle |
| $D_i$ | The computation quantity of user $i$ |
| $F_i^g$ | The amount of resources allocated by the BS $g$ to user $i$ |
| $f_i^j$ | The amount of resources allocated by the UAV $j$ to user $i$ |
| $M_i^g$ | Resource pricing of BS $g$ to user $i$ |
| $M_j^g$ | Pricing of UAV $j$ employed by BS $g$ |
| $m_i^j$ | Resource pricing of UAV $j$ to user $i$ |
| $Need_i^g$ | Whether user $i$ chooses to offload to BS $g$ |
| $Need_i^j$ | Whether user $i$ chooses to offload to UAV $j$ |
| $Role_i^j$ | Whether UAV $j$ is a relay when dealing with the task of user $i$ |

## References

1. Kitae, K.; Hong, C.S. Optimal Task-UAV-Edge Matching for Computation Offloading in UAV Assisted Mobile Edge Computing. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 18–20 September 2019; pp. 1–4. [CrossRef]
2. Alia, A.; Niyato, D. Hierarchical Game-Theoretic and Reinforcement Learning Framework for Computational Offloading in UAV-Enabled Mobile Edge Computing Networks With Multiple Service Providers. *IEEE Internet Things J.* **2019**, *6*, 8753–8769. [CrossRef]
3. Wenjie, W.; Wei, Z. Computational offloading with delay and capacity constraints in mobile edge. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6. [CrossRef]
4. Yi, L.; Shengli, X.; Yan, Z. Cooperative Offloading and Resource Management for UAV-Enabled Mobile Edge Computing in Power IoT System. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12229–12239. [CrossRef]
5. Tiankui, Z.; Yu, X.; Jonathan, L.; Dingcheng, Y.; Lin, X. Joint Computation and Communication Design for UAV-Assisted Mobile Edge Computing in IoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5505–5516. [CrossRef]
6. Tong, Z.; Ye, F.; Yan, M.; Liu, H.; Basodi, S. A Survey on Algorithms for Intelligent Computing and Smart City Applications. *Big Data Min. Anal.* **2021**, *4*, 18. [CrossRef]
7. Mitsis, G.; Tsiropoulou, E.E.; Papavassiliou, S. Data Offloading in UAV-Assisted Multi-Access Edge Computing Systems: A Resource-Based Pricing and User Risk-Awareness Approach. *Sensors* **2020**, *20*, 2434. [CrossRef]
8. McNabbo, M. Drones and Mobile Edge Computing: Verizon and AWS Expand Service. 2022. Available online: https://dronelife.com/2022/01/23/drones-and-mobile-edge-computing/ (accessed on 21 April 2022).
9. Jiaxin, C.; Ping, C.; Qihui, W.; Yuhua, X.; Nan, Q.; Tao, F. A game-theoretic perspective on resource management for large-scale UAV communication networks. *China Commun.* **2021**, *18*, 70–87. [CrossRef]

10. Xintong, Q.; Zhengyu, S.; Yuanyuan, H.; Xin, S. Joint Resource Allocation and Trajectory Optimization for Multi-UAV-Assisted Multi-Access Mobile Edge Computing. *IEEE Wirel. Commun. Lett.* **2021**, *10*, 1400–1404. [CrossRef]
11. Wen'An, Z.; Lixia, L.; Jianlong, L.; Donglong, Z.; Yifan, X. Joint Offloading Decision and Resource Allocation for Multiuser NOMA-MEC Systems. *IEEE Access* **2019**, *7*, 181100–181116. [CrossRef]
12. Bi, R.; Liu, Q.; Ren, J.; Tan, G. Utility Aware Offloading for Mobile-Edge Computing. *Tsinghua Sci. Technol.* **2021**, *v.26*, 107–118. [CrossRef]
13. Xu, X.; Li, H.; Xu, W.; Liu, Z.; Yao, L.; Dai, F. Artificial Intelligence for Edge Service Optimization in Internet of Vehicles: A Survey. *Tsinghua Sci. Technol.* **2022**, *27*, 270–287. [CrossRef]
14. Long, L.; Zichen, L.; Jinglin, S.; Yiqing, Z.; Dawei, Q.; Shunqing, X. Joint optimization strategy of computation offloading and resource allocation in mobile edge computing. *Chin. High Technol. Lett.* **2020**, *30*, 9.
15. Heyu, L.; Yitao, X.; Dianxiong, L.; Zhiyong, D.; Huiming, Q.; Xiaodu, L.; Xiaobing, T. Energy Efficient Task Cooperation for Multi-UAV Networks: A Coalition Formation Game Approach. *IEEE Access* **2020**, *8*, 149372–149384. [CrossRef]
16. Ying, C.; Shuang, C.; Bilian, W.; Xin, C. Cost-efficient computation offloading in UAV-enabled edge computing. *IET Commun.* **2020**, *14*, 2462–2471.
17. Yanling, R.; Zhibin, X.; Zhenfeng, D.; Xiyuan, S.; Jie, X.; Yubo, T. Computation offloading game in multiple unmanned aerial vehicle-enabled mobile edge computing networks. *IET Commun.* **2021**, *15*, 1392–1401.
18. Alioua, A.; Djeghri, H.E.; Cherif, M.; Senouci, S.M.; Sedjlmaci, H. UAVs for Traffic Monitoring: A Sequential Game-based Computation Offloading/Sharing Approach. *Comput. Netw.* **2020**, *177*, 107273. [CrossRef]
19. Ei, N.N.; Kang, S.W.; Alsenwi, M.; Tun, Y.K.; Hong, C.S. Multi-UAV-Assisted MEC System: Joint Association and Resource Management Framework. In Proceedings of the 2021 International Conference on Information Networking (ICOIN). IEEE, Jeju, Korea, 13–16 January 2021; pp. 213–218.
20. Yan, C.; Ye, Y. A Hierarchical Game Optimization Method for UAVs and Users in MEC System. *Commun. Technol.* **2020**, *53*, 6.
21. Chao, D.; Yun, S.; Yuben, Q. A survey of UAV-based edge intelligent computing. *Chin. J. Intell. Sci. Technol.* **2020**, *2*, 227–239. [CrossRef]
22. Jia, L.; Xu, Y.; Sun, Y.; Feng, S.; Anpalagan, A. Stackelberg Game Approaches for Anti-Jamming Defence in Wireless Networks. *IEEE Wirel. Commun.* **2018**, *25*, 120–128. [CrossRef]
23. Messous, M.A.; Senouci, S.M.; Sedjelmaci, H.; Cherkaoui, S. A Game Theory Based Efficient Computation Offloading in an UAV Network. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4964–4974. [CrossRef]
24. Feng, Z.; Ren, G.; Chen, J.; Zhang, X.; Luo, Y.; Wang, M.; Xu, Y. Power Control in Relay-Assisted Anti-Jamming Systems: A Bayesian Three-Layer Stackelberg Game Approach. *IEEE Access* **2019**, *7*, 14623–14636. [CrossRef]
25. Zhang, K.; Mao, Y.; Leng, S.; Maharjan, S.; Zhang, Y. Optimal delay constrained offloading for vehicular edge computing networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6. [CrossRef]
26. Xu, Y.; Chen, J.; Xu, Y.; Gu, F.; Yao, K.; Jia, L.; Liu, D.; Wang, X. Energy-Efficient Channel Access and Data Offloading Against Dynamic Jamming Attacks. *IEEE Trans. Green Commun. Netw.* **2021**, *5*, 1734–1746. [CrossRef]
27. Yang, L.; Xu, C.; Zhan, Y.; Liu, Z.; Guan, J.; Zhang, H. Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach. *Comput. Netw.* **2017**, *129*, 399–409.
28. Wang, T.; Sun, Q. Stackelberg Game based Computation Offloading and Resource Allocation in Mobile Edge Computing. In Proceedings of the 2020 International Conference on Space-Air-Ground Computing (SAGC), Beijing, China, 4–6 December 2020.
29. Songdi, Q.; The Textbook Editorial Committee of Operations Research. *Operations Research*, 4th ed.; Tsinghua University Press: Beijing, China, 2012; pp. 15–28.