electronics

MDPI

Article

Dynamic Task Migration Combining Energy Efficiency and Load Balancing Optimization in Three-Tier UAV-Enabled Mobile Edge Computing System

Wu Ouyang †, Zhigang Chen *,†, Jia Wu *,†, Genghua Yu † and Heng Zhang †

School of Computer Science and Engineering, Central South University, Changsha 410083, China; ouyangwu@csu.edu.cn (W.O.); cnygh821@163.com (G.Y.); zhangheng1018@csu.edu.cn (H.Z.);

* Correspondence: czg@csu.edu.cn (Z.C.); jiawu0510@csu.edu.cn (J.W.); Tel.: +86-133-8748-0797 (Z.C.); +86-182-7312-5752 (J.W.)

† These authors contributed equally to this work.

Abstract: As transportation becomes more convenient and efficient, users move faster and faster. When a user leaves the service range of the original edge server, the original edge server needs to migrate the tasks offloaded by the user to other edge servers. An effective task migration strategy needs to fully consider the location of users, the load status of edge servers, and energy consumption, which make designing an effective task migration strategy a challenge. In this paper, we innovatively proposed a mobile edge computing (MEC) system architecture consisting of multiple smart mobile devices (SMDs), multiple unmanned aerial vehicle (UAV), and a base station (BS). Moreover, we establish the model of the Markov decision process with unknown rewards (MDPUR) based on the traditional Markov decision process (MDP), which comprehensively considers the three aspects of the migration distance, the residual energy status of the UAVs, and the load status of the UAVs. Based on the MDPUR model, we propose a advantage-based value iteration (ABVI) algorithm to obtain the effective task migration strategy, which can help the UAV group to achieve load balancing and reduce the total energy consumption of the UAV group under the premise of ensuring user service quality. Finally, the results of simulation experiments show that the ABVI algorithm is effective. In particular, the ABVI algorithm has better performance than the traditional value iterative algorithm. And in a dynamic environment, the ABVI algorithm is also very robust.

Keywords: mobile edge computing (MEC); unmanned aerial vehicle (UAV); task migration; energy-efficient; load balancing; Markov decision process (MDP)

Citation: Ouyang, W.; Chen, Z.; Wu, J.; Yu, G.; Zhang, H. Dynamic Task Migration Combining Energy Efficiency and Load Balancing Optimization in Three-Tier UAV-Enabled Mobile Edge Computing System. *Electronics* **2021**, *10*, 190. https://doi.org/10.3390/electronics10020190

Received: 20 November 2020
Accepted: 12 January 2021
Published: 15 January 2021

1. Introduction

Nowadays, with the rapid development of smart mobile devices (SMDs) and online applications, users' requirements for services is increasing. In the real scene, while SMDs are portable, and with limited computation and storage capability. To solve this problem, mobile edge computing (MEC) is proposed, in which users can offload their task to the nearby devices which have strong computation and storage capability, thereby reducing the delay and energy consumption of SMDs [1]. However, the service scope of a single edge server is limited . When a SMD deviates from the service range of its original edge server, it is hard to feedback the results to the SMD [2].

To guarantee the service quality of users, task migration has become a very promising method. Specifically, task migration means that the original edge server migrates the tasks offloaded by users to other edge servers to ensure the service quality of users and reduce the energy consumption of SMDs. However, task migration may cause service interruption or additional network overhead . Therefore, the quality of the task migration strategy will directly affect the service quality of users, the load status of the edge servers, and the energy consumption generated by the task migration. At present, some researchers focus

on the cost of task migration [3], or only on the service quality of users [4]. But few people consider the load of edge servers. As far as we know, the load of the edge servers will affect the operating efficiency of the program and the service life of the edge servers. Therefore, the research direction of this paper is not limited to one aspect, but considers the following three aspects comprehensively, namely the cost of task migration, the load status of the edge servers and the quality service of the users. The purpose of this paper is to design an efficient task migration strategy to achieve load balancing and reduce the total energy consumption of the unmanned aerial vehicle (UAV) group under the premise of ensuring user service quality.

In the process of designing an efficient task migration strategy, it is necessary to consider a three-layer MEC system composed of SMDs, a UAV group, and a base station (BS). In such a three-layer MEC system, the UAVs in the UAV group act as dynamic edge servers [5], while the BS acts as a static edge server [6]. The advantages of the UAVs as dynamic edge servers are twofold. First, compared with the traditional cellular infrastructure communication MEC, the communication technology between UAVs has received more attention, and new research hotspots are constantly emerging [7]; second, UAVs have the advantages of high maneuverability, fast mobility, and unrestricted geographic location, which make up for the shortcomings of static edge servers with a fixed geographic location and no flexibility. At the same time, the BS as the static edge server can greatly alleviate the limited computing and storage capabilities of the UAVs. Therefore, the UAV group and the BS can provide users with more stable and flexible services [8]. In addition, the task migration strategy focuses on the migration decision. When a UAV in the UAV group receives the computing task offloaded by a SMD, the UAV will first decide on whether to migrate based on the computing resources required by the task. When the computing resources required for the task are too large and the current UAV is unable to handle it, the current UAV will directly migrate the task to other UAVs or the BS for processing. It also needs to be considered that in real scenes, the energy of the UAV is limited. When the energy of the UAV is too low, it will withdraw from the UAV group and cannot provide services to users. Therefore, in this paper, an efficient task migration strategy is designed in a three-layer MEC system composed of SMDs, a UAV group, and a BS. Moreover, the task migration strategy can help the UAV group achieve load balancing and reduce the total energy consumption of the UAV group on the premise of ensuring user service quality.In general, we select multiple migration targets and then calculate the migration strategy. From these alternative migration strategies, we select the optimal migration strategy.

In previous studies, it has been proved that Markov decision process (MDP) can be used to solve the task migration problem [9]. Based on Markov decision process (MDP), we established the Markov decision process with unknown rewards (MDPUR) model, in which the return function fully considers the load state and residual energy state of the UAVs. In addition, we also designed the advantage-based value iteration (ABVI) algorithm combining the parent-generation crossover (PC) algorithm to get the optimal migration strategy. Compared with the traditional value iteration algorithm, the ABVI algorithm has better performance, which not only ensures the service quality of users but also helps the UAV group to achieve load balancing, improve the flight time of the UAV group and reduce the total energy consumption of the UAV group. The main contributions of this paper are as follows.

- This paper combines dynamic edge servers (i.e., UAVs) with a static edge server (a BS) for the first time and establishes a three-layer MEC system consisting of SMDs, a UAV group, and a BS. Meanwhile, the communication model, delay model, and energy consumption model based on the MEC system are established.
- We designed the ABVI algorithm combining the PC algorithm, which can reduce the total number of value iterations, improve the efficiency of obtaining the optimal migration strategy, and further ensure the service quality of users.
- We establish the MDPUR model, in which the return function r is very innovative and dynamically adjusted according to the load state and residual energy state of the

UAVs, which makes the ABVI algorithm very robust. In addition, each UAV in the MDPUR model is equivalent to a node on the migration path, and the state of the node is defined as dynamically changing, which is more in line with the UAV-enabled MEC scene. Meanwhile, the validity of the ABVI algorithm and its feasibility in actual scenes are proved.

- In the process of solving the optimal task migration strategy, we consider advantage vector. In a given initial state, each migration action will change the state of the UAVs. At the same time, in the iteration process of the ABVI algorithm, we will classify the value function vectors, which is beneficial to improve the accuracy of solving the optimal migration strategy. In particular, we are not limited to a single target node, but solve multiple migration strategies through the ABVI algorithm, which greatly enhanced the reliability of task migration.

The rest of this paper is shown below. Section 2 introduces the related work in this research field. In Section 3, the system model, MDPUR model and problem formulation are described. Then, in Section 4, the ABVI algorithm and the PC algorithm are proposed. Also, we analyze the results of the simulation experiment in Section 5. Finally, Section 6 summarizes the paper.

2. Related Work

At present, to improve the performance of the MEC system, more and more researchers have introduced UAV into a MEC system, established a UAV-enabled MEC system, and launched a series of research work from the perspective of task migration. In the following, we introduce the research status in terms of task migration and UAV-enabled communication. The existing task migration work is mainly focused on reducing delay and migration costs [10]. In References [11,12], researchers use the MDP framework to predict the users' possible mobile location so that the obtained migration decision can reduce the delay of task transmission. In Reference [13], S. Wang et al. propose a prediction module that can predict the future cost of each user, which can help the system calculate the task migration decision. In addition, A. Ceselli et al. apply deep learning techniques to the prediction of user mobility [14]. Although these methods are very effective in simulations, they are difficult to apply to actual scenarios because they take a long time to collect user information. In contrast, the proposed method in our paper only needs to know the status of the edge servers and the location of the users in each time period.

At the same time, a considerable part of the research work does not start from predicting user movement. In Reference [15], Liu et al. propose an efficient one-dimensional search algorithm, which can obtain the migration strategy with the lowest average delay. In Reference [16], the researchers consider the impact of latency and reliability on user service quality and propose a new type of network design. Moreover, this paper restricts the types of tasks, and not all tasks can be migrated. Besides, Liqing Liu et al. consider both task migration delay and migration cost, and finally construct a multi-objective optimization problem [17]. However, the above papers mainly focus on latency and ignore other indicators. From the related papers, we found that the load status of the edge servers also directly affects the service quality of users [18]. When the load of the edge servers is too high, the task processing delay will also increase accordingly, which will affect the service quality of users. Therefore, it is necessary to consider the load status of the edge servers [19]. At present, few researchers consider the load status of the edge servers, the service quality of users, and energy consumption at the same time.

As for the research of MDP-based migration decision, W. Zhang et al. propose an MDP model based on the status of BSs and edge servers, emphasizing the importance of reducing the complexity of the algorithm [20]. As far as we know, there are three main types of MDP-based migration strategies [21]. The one is a migration strategy based on reinforcement learning, which changes the reward function in the model by observing the users' behavior, and then solves the migration path [22]; the other is the optimal migration strategy based on iteration algorithms such as value iteration or strategy iteration [23].

Although this kind of strategy was first proposed, it has the disadvantage of too high complexity; the third is the migration strategy based on the maximum and minimum regret method [24], this type of migration strategy is classified and sorted under the condition that all candidate value function sets can be calculated, and then optimize the "minimax regret" to find the best strategy [25,26]. However, the convergence speed of the algorithm proposed in these works needs to be improved, and the return function should also be more in line with the actual scenario.

In order to provide users with flexible services, researchers begin to use UAVs as edge servers. In recent years, many researchers have focused on the safety of communication between UAVs and optimizing the flight path of UAVs. In Reference [27], a UAV communication system based on 5G technology is proposed to enhance the network coverage of the UAVs and improve the communication security between UAVs. In Reference [28], a new data distribution model is designed by the researchers to improve the anti-jamming transmission capability of the UAVs, which can ensure communication security. In addition, Jingjing Gu et al. propose a new algorithm for mobile users to plan the flight path of the UAVs to ensure the service quality of users [29]. And an online learning algorithm is proposed in Reference [30], which optimizes the flight path of the UAVs according to the distribution state of users to improve throughput. But most of the works do not take into account the energy consumption of UAVs, which will cause the UAVs to have a shorter flight time, which ultimately affects the service quality of users.

3. System Model

3.1. Overall Architecture

As shown in Figure 1, we considered a multi-user multi-edge server MEC system with dynamic task migration and a combination of dynamic and static edge servers. The system consists of N smart mobile devices (SMD), M four-axis unmanned aerial vehicle (UAV) and a single base station (BS). Here, each UAV has the functions of hovering and flying, and UAVs act as dynamic edge servers to provide SMDs with no geographical restrictions and more flexible services, while BS as a static edge server has stronger computing power and can provide more stable and lasting services for SMDs. Specifically, we use $\mathcal{N} = \{1, 2, ..., N\}, \forall\, i \in \mathcal{N}$ and $\mathcal{M} = \{1, 2, ..., M\}, \forall\, j \in \mathcal{M}$ to denote the SMD set and UAV set, respectively. In this paper, the total time T for task migration completion is divided into K time slots on average, and the length of each time slot t is τ, that is, $T = K \cdot \tau$. Therefore, we define $\mathcal{T} = \{1, 2, ..., K\}, \forall\, t \in \mathcal{T}$ to represent the set of time slots. In Figure 1, N SMDs are randomly distributed in an area. In order to specify the positions of UAV and SMD, we randomly set the origin within the area, ie (0,0,0), and establish a spatial rectangular coordinate system. Next, we define the triplet $\boldsymbol{l}^t_{SMD,i} = (x^t_i, y^t_i, 0), \forall i \in \mathcal{N}$ to represent the geographic location of the SMD i at time slot t, where x^t_j and y^t_j represent the distance of the SMD i from the origin in the x-axis and y-axis directions at time slot t, respectively. Similarly, we use $\boldsymbol{l}^t_{UAV,j} = (\widetilde{x}^t_j, \widetilde{y}^t_j, H^t_j), \forall j \in \mathcal{M}$ to denote the geographic location of UAV j at time slot t, where $\widetilde{x}^t_j$ and $\widetilde{y}^t_j$ denote the distance of UAV j from the origin in the x-axis and y-axis directions at time slot t, and H^t_j denotes the height of UAV j at time slot t, that is, the distance from the origin in the z-axis direction of time slot t. Because the location of the base station is fixed and will not change with time, we define $\boldsymbol{l}_{BS} = (x_{BS}, y_{BS}, 0)$ to represent the geographic location of the base station in the current area of the system. For ease of understanding, the main symbols involved in this paper are defined in Table 1.

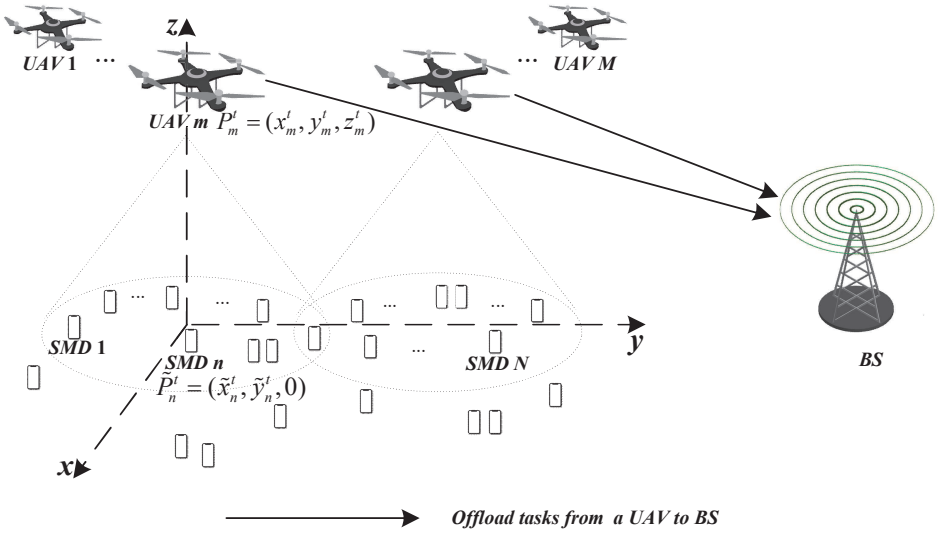

Figure 1. Overall architecture of the system.

Table 1. Definition of main symbols.

Symbols	Definition
$\mathcal{N}$	The set of SMDs
$\mathcal{M}$	The set of UAVs
$\mathcal{T}$	The set of time slots
$l_{SMD,i}^t$	The geographic location of the SMD i at time slot t
$l_{UAV,j}^t$	The geographic location of UAV j at time slot t
l_{BS}	The geographic location of the base station in the current area of the system
$\widetilde{S}_j^t$	The maximum service range of UAV j at time slot t
H_j^t	The flying height of UAV j at time slot t
U_i	The task offloaded from SMD i to the UAV group at $t = 0$
F_i	The total number of CPU cycles required by the computing task offloaded by SMD i
D_i	The size of tasks offloaded from SMD i to the UAV group
$T_i^{deadline}$	The maximum deadline for completing the migration of task offloaded by SMD i
$h_{i,j}^t$	The channel gains of SMD i and UAV j in time slot t
$d_{i,j}^t$	The distance between SMD i and UAV j in time slot t
$r_{i,j}^t$	The uplink transmission power of SMD i to offload tasks to UAV j in time slot t
$p_{i,j}^t$	The transmission power of SMD i offload task to UAV j in time slot t
$h_{j,BS}^t$	The channel gain between UAV j and the BS in time slot t
$d_{j,BS}^t$	The distance between SMD j and BS in time slot t
$r_{j,BS}^t$	The transmission rate of UAV j transfer the task to the BS at time slot t
$p_{j,BS}^t$	The transmission power of UAV j transfer the task to the BS at time slot t
$f_{i,j}$	The computing resources allocated to SMD i by UAV j
$T_{i,j}^{pro}$	The time required by UAV j to process task U_i
$T_{i,j}^{mig}$	The time for UAV j to migrate task U_i
L_j^t	The length of the task queue of UAV j at time slot t
$E_{com,j}^t$	The computed energy consumption of UAV j in time slot t
$E_{mig,j}^t$	The energy consumption of UAV j migrate tasks to other UAVs or the BS in time slot t
$E_{total,j}^t$	The total energy consumption of UAV j in time slot t

To strengthen the communication capability between UAVs and SMDs on the ground, current UAVs will install a directional antenna. Since the direction of a UAV directional antenna has a certain angle with the vertical direction, the range of services provided by the UAV is related to its height. As shown in Figure 2, we assume that the angle between

the directional antenna of UAV j and the vertical direction is θ, that is, the antenna angle of UAV j is θ. Also, the service range of each UAV is a circle. When the antenna angle of a UAV is fixed, the service range of UAV expands as the UAV height increases; otherwise, the service range of the UAV shrinks as the UAV height decreases. Specifically, under the premise of knowing the flying height H_j^t of UAV j at time slot t, we can obtain the maximum service range $\widetilde{S}_j^t, \forall j \in \mathcal{M}$ of UAV j at time slot t by Formula (1), and the unit of service range is square meters.

$$\widetilde{S}_j^t \ = \ (H_j^t \cdot \tan\theta)^2 \cdot \pi, \tag{1}$$

where π is the PI. At the same time, the speed of information transmission between SMDs and UAVs also slows down as the distance increases. In short, to maximize the performance of the system and avoid the service quality affected caused by the UAV being too far from the ground, the UAV at any time slot should not exceed the maximum height H^{max} preset by the system. Then, the flight height H_j^t of UAV j at time slot t naturally meets the following restrictions, namely,

$$H_j^t \ \leq \ H^{max}, \forall j \in \mathcal{M}, \forall t \in \mathcal{T}. \tag{2}$$

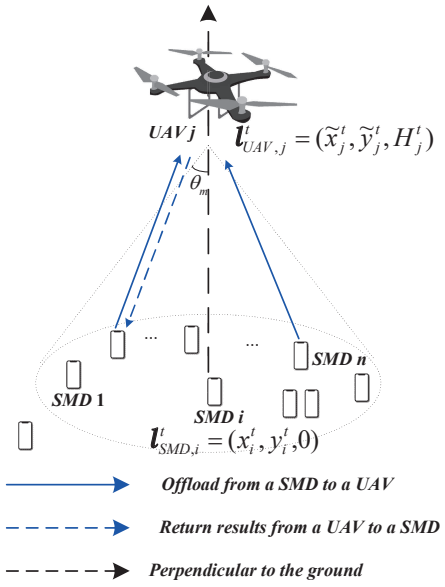

Figure 2. The service range of unmanned aerial vehicle (UAV) j.

In this paper, after a SMD offloads the task to a UAV, the UAV will process the task immediately. When the SMD deviates from the service scope of the UAV, on the one hand, the UAV may transfer the task offloaded by the SMD to another UAV for processing to ensure the continuous service for SMD; on the other hand, the UAV may also directly transfer the task to the BS in the current area of the system. The reason why the task is directly transmitted to the base station by the UAV is that the computing power required to process the task offloaded from SMD to UAV far exceeds the maximum computing power of every UAV, the BS as a static edge server has stronger computing power than the dynamic edge server such as UAV. We describe the above two processes of transferring the task from the UAV to another UAV and from the UAV to the BS as a task migration process.

Moreover, the main content of our research is to improve the efficiency of task migration in the MEC system. Therefore, we do not consider the collaborative capabilities of UAV groups, that is, we consider that the tasks offloaded to UAVs by SMDs are inseparable. Similar to Reference [31], we do not distinguish between types of tasks. We assume that

only one task is offloaded to the UAV group by SMD i at $t = 0$, and the task is defined as a triple, that is $U_i = (F_i, D_i, T_i^{deadline})$, where F_i represents the total number of CPU cycles required by the computing task offloaded by SMD i. D_i indicates the size of tasks offloaded from SMD i to the UAV group, and the unit is bit. As for $T_i^{deadline}$, it is the maximum deadline for completing the migration of tasks offloaded by SMD i. According to Reference [32], as long as the task migration occurs in the MEC system, there will be a "migration time", that is, the delay caused by the task migration. It can be seen that SMDs will not be able to get a more timely response to task processing results due to task migration, so reducing migration time is critical. After completing the overall description of the system, we begin to introduce the corresponding communication model of the system in the next section.

3.2. Communication Model

In this section, we introduce the communication model, which consists of three parts, namely the communication between SMDs and UAVs, the communication between UAVs and other UAVs and BS, and the communication between BS and SMDs. Obviously, the three parts of the communication are all wireless rather than wired communication, so the efficiency of task transmission is mainly affected by the three aspects of distance, noise power and network bandwidth.

For the communication between SMDs and UAVs, it mainly means that SMDs in the system offload the corresponding computation tasks to the UAVs, and UAVs return the result of the tasks computation to the corresponding SMDs. Moreover, multiple SMDs can simultaneously offload tasks to the same UAV. According to Reference [33], we can know that the wireless channels between SMDs and UAVs are determined by the location of the SMD links. Therefore, the channel gains $h_{i,j}^t$ of SMD i and UAV j in time slot t is as follows,

$$h_{i,j}^t = \frac{h_0}{(d_{i,j}^t)^2}, \tag{3}$$

where h_0 represents the channel gain between a SMD and a UAV when the distance is one meter, and $d_{i,j}^t$ represents the distance between SMD i and UAV j in time slot t. For simplicity, we directly use the Euclidean distance to define $d_{i,j}^t$, that is,

$$d_{i,j}^t = \sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 + (H_j^t)^2}. \tag{4}$$

Also, regarding the transmission rate between SMDs and UAVs, according to Reference [34], the rate of wireless transmission is mainly determined by four aspects: network bandwidth, channel gain, transmission power, and noise power. We define the transmission rate of SMD i offload tasks to UAV j in time slot t, that is,

$$r_{i,j}^t = B_{i,j} \cdot log_2(1 + \frac{p_{i,j}^t \cdot h_{i,j}^t}{\sigma_{SMD}^2}), \tag{5}$$

where $B_{i,j}$ represents the channel bandwidth between SMD i and UAV j, and σ_{SMD}^2 represents the noise power between SMDs and UAV group. $p_{i,j}^t$ represents the transmission power of SMD i offload task to UAV j at time slot t. In particular, in real application scenarios, SMDs rely on battery power, which means that the energy of SMDs is limited. In addition, the transmission power mentioned in the previous content will directly affect the residual energy of the battery, so we set the maximum transmission power of SMD p_{SMD}^{max}. So $p_{i,j}^t$ satisfy the following inequality,

$$0 \leq p_{i,j}^t \leq p_{SMD}^{max}, i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{6}$$

For the communication between UAV and other UAVs and BS, it mainly means that UAVs can transfer the computation tasks that cannot be processed to other UAVs or BS, and

multiple UAVs can also transfer tasks to the same UAV or BS at the same time. It should be noted that since the task migration between UAV and the task migration between UAV and BS involves the same communication principle, we only describe the communication process between UAVs and BS. Similar to the communication process in the first part, we use $h^t_{j,BS}$ to represent the channel gain between UAV j and the BS. The specific formula is as follows,

$$h^t_{j,BS} = \frac{\widetilde{h}_0}{(d^t_{i,BS})^2}, \tag{7}$$

where $\widetilde{h}_0$ represents the channel gain between UAV and the BS when the distance is one meter. $d^t_{j,BS}$ represents the distance between SMD j and BS in time slot t, and the specific definition is as follows,

$$d^t_{j,BS} = \sqrt{(x^t_j - x^t_{BS})^2 + (y^t_j - y^t_{BS})^2 + (H^t_j)^2}. \tag{8}$$

In addition, the transmission rate and the constraint of the transmission power of UAV j transfer the task to the BS at time slot t are shown in Formulas (9) and (10), respectively.

$$r^t_{j,BS} = B_{j,BS} \cdot log_2(1 + \frac{p^t_{j,BS} \cdot h^t_{j,BS}}{\sigma^2_{UAV}}), \tag{9}$$

$$0 \leq p^t_{j,BS} \leq p^{max}_{UAV}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{10}$$

where $B_{j,BS}$ and σ^2_{UAV} are the channel bandwidth between UAV j and the BS and the noise power between the UAV group and the BS, respectively. $P^m_{UAV}ax$ is the maximum transmission power of the UAVs in the system. In addition, the transmission rate and transmission power of task migration between UAVs are defined as r^t_{UAV} and p^t_{UAV}, respectively. And r^t_{UAV} and p^t_{UAV} satisfy $r^t_{UAV} = r^t_{j,BS}$ and $p^t_{UAV} = p^t_{j,BS}$, respectively.

3.3. Delay Model

In this section, we introduce the delay model of the system. Before introducing in detail, we first describe the specific process of task migration. Figure 3 shows the specific process of task migration by UAV j, where $f_{i,j}$ represents the computing resource allocated to SMD i by UAV j. And the meaning of $\frac{F_i}{f_{i,j}}$ is the time required for UAV j to process task U_i offloaded by SMD i. With the rapid growth of the number of real-time applications in SMDs, the length of time the edge server spends processing tasks will seriously affect the service quality of the users. We did a survey, the survey results show that: when the threshold is greater than 1.2 s, the number of users is the highest, which users believe has seriously affected the service quality. Therefore, we assume that once UAV j processes a task for more than 1.2 s, it is considered that the computation power required by the task exceeds the maximum computation power of UAV j. Whether the time spent by UAV j in processing tasks exceeds 1.2 s corresponds to two cases. Case 1, When UAV j processes a task for more than 1.2 s, to ensure that the task can be processed smoothly and provide good service for SMD i, UAV j needs to migrate the task to another UAV or the BS, and then rely on the another UAV that still has strong computing power or the BS with stronger computation power than UAVs to handles task U_i. Case 2, when UAV j does not spend more than 1.2 s processing task U_i, UAV j still handle task U_i offloaded by SMD i.

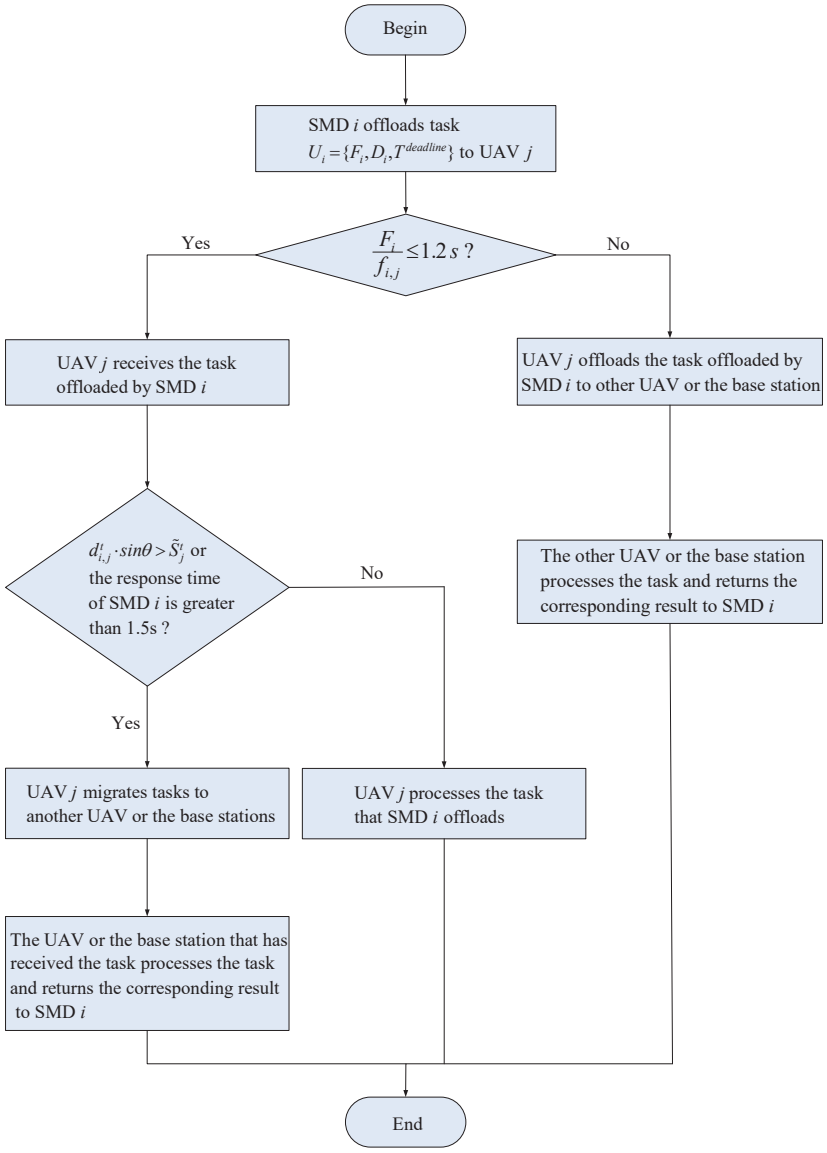

Figure 3. Service range of UAV j.

Under case 2, to scientifically and efficiently determine the task migration timing of UAV j, we define the following two task migration triggering conditions. One of the triggering conditions is $d_{i,j}^t \cdot \sin\theta > \widetilde{S}_j^t$, which indicates that SMD i has left the service range of UAV j at time slot t. At this time, to continue to provide services to SMD i, UAV j needs to migrate the task U_i that has been offloaded onto it. Another trigger condition is based on the focus on the quality of user service in this paper. Specifically, each UAV in the system will regularly send a very small data packet to the SMDs within its service range. After receiving the data packet, the SMDs will send a response to the UAV. Through the above operations, we can calculate the "response time" of SMDs. We believe that if the "response time" of SMD i is greater than 1.5 s, it indicates that the service quality of SMD i is not good, and task U_i in UAV j needs to be migrated. In summary, as long as one of the above two conditions is met, task migration is required.

When SMD i offloads task U_i to UAV j and the computing resources required by the task do not exceed the maximum computing power of UAV j, UAV j allocates the corresponding computing resources to SMD i and starts processing task U_i. In the previous

content, we have defined $f_{i,j}$ to represent the computing resources allocated to SMD i by UAV j , so the time required by UAV j to process task U_i is defined as

$$T_{i,j}^{pro} = \frac{F_i}{f_{i,j}}. \tag{11}$$

When task migration conditions in Figure 3 are met, UAV j will migrate the task to another UAV or the BS. Therefore, the time for UAV j to migrate task U_i is defined as follows,

$$T_{i,j}^{mig} = \frac{D_i}{\frac{1}{K} \cdot \sum_{t=0}^{K-1} r_{j,BS}^t} = \frac{D_i}{\frac{1}{K} \cdot \sum_{t=0}^{K-1} r_{UAV}^t} = \frac{K \cdot D_i}{\sum_{t=0}^{K-1} r_{UAV}^t}. \tag{12}$$

Since the transmission speed changes with time, we use the average transmission speed for calculation. Also, $T_{i,j}^{mig}$ also needs to meet the following restrictions, that is,

$$T_{i,j}^{mig} \leq T_j^{deadline}, i \in \mathcal{N}, j \in \mathcal{M}. \tag{13}$$

In other words, in the process of task migration, we must ensure that the migration time of task U_i is transferred to another UAV or the BS cannot exceed $T_i^{deadline}$, so as to guarantee the service quality of the users.

From the previous content, we know that UAV may receive multiple task requests from multiple SMDs at the same time. Here, we assume that the process of offloading tasks from SMDs is continuous. Therefore, we use L_j^t to denote the length of the task queue of UAV j at time slot t, and the specific definition of L_j^t is as follows,

$$L_j^t = \sum_{i=1}^{N} \int_0^t r_{i,j}^t dt - \int_0^t \mu_j^t dt + \delta_{UAV,j}, \tag{14}$$

where $\delta_{UAV,j}$ represents the program that UAV j must execute to maintain the normal operation of UAV j, such as the operating system. As for μ_j^t, it is the total number of UAV j processing tasks in time slot t, in bits. And μ_j^t satisfies the following equation, that is,

$$\mu_j^t = \sum_{i=1}^{N} \frac{f_{i,j} \cdot \tau}{c}, \tag{15}$$

where c is the CPU cycles required by a UAV to handle any type of computing task.

Although UAVs have much more computing power than SMDs, their computing and storage capabilities are also limited. Therefore, $f_{i,j}$ and L_j^t satisfy inequality (16) and inequality (17), respectively.

$$f_{i,j}^t \leq f_{UAV}^{max}, i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{16}$$

where f_{UAV}^{max} is the maximum computing power of UAV.

$$L_j^t \leq L_{UAV}^{max}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{17}$$

where L_{UAV}^{max} is the maximum storage space corresponding to the task queue of UAV.

In addition to some delay overhead, the system also has energy consumption overhead. Therefore, we introduce the energy consumption model of the system in detail below.

3.4. Energy Consumption Model

In this section, we introduce the energy consumption model of the system. We focus on the energy consumption of three parts, namely the flight energy consumption of UAVs, the computing energy consumption of UAVs, and task migration energy consumption of UAVs.

For flight energy consumption of UAVs, according to Reference [35], it can be known that the flight energy consumption of a UAV is mainly determined by the flight speed, acceleration, and altitude of the UAV. The flight speed v_j^t (m/s) and acceleration a_j^t (m/s^2) of UAV j at time slot t are defined as follows, respectively.

$$v_j^t = \frac{\|\boldsymbol{l}_{UAV,j}^t - \boldsymbol{l}_{UAV}^{t-1}\|}{\tau}, \tag{18}$$

$$a_j^t = \frac{v_j^t - v_j^{t-1}}{\tau}. \tag{19}$$

Therefore, we can get the flight energy consumption of UAV j during time slot t, that is,

$$E_{fly,j}^t = k_1\|v_j^t\|^3 + \frac{k_2}{\|v_j^t\|}(1 + \frac{\|a_j^t\|^2}{g^2}) + k_3 \cdot \|H_j^t\|^2, \tag{20}$$

where k_1, k_2 and k_3 satisfy Equations (21)–(23), respectively.

$$k_1 = 0.5 \cdot \rho \cdot \widetilde{M}_j \cdot \tau, \tag{21}$$

where ρ (kg/m^3) represents the air density in the flying area of the UAVs, and $\widetilde{M}_j$ is the mass of UAV j.

$$k_2 = \frac{2 \cdot (\widetilde{M}_j)^2 \cdot \tau}{\pi \cdot w^3 \cdot c_j}, \tag{22}$$

where π stands for PI, and w (m/s) is wind speed. As for c_j, it is the number of revolutions per second of the propeller of UAV j.

$$k_3 = \frac{g \cdot \tau}{2}, \tag{23}$$

where g is the acceleration of gravity.

Besides, during the flight of the UAVs, we also restrict the speed, acceleration, and trajectory of the UAVs, and these restrictions corresponding to inequality (24), inequality (25), and inequality group (26), respectively.

$$\|v_j^t\| \leq v^{max}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{24}$$

$$\|a_j^t\| \leq a^{max}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{25}$$

$$\begin{cases} \sum\limits_{t=0}^{K-1} \|\boldsymbol{l}_j^t - \boldsymbol{l}_j^{t-1}\| \leq \eta, \ j \in \mathcal{M}, t \in \mathcal{T}, \\ \boldsymbol{l}_{UAV,j}^0 = \boldsymbol{l}_{UAV}^{initial}, \ \boldsymbol{l}_{UAV}^{K-1} = \boldsymbol{l}_{UAV}^{end}. \end{cases} \tag{26}$$

Among them, v^{max} and a^{max} are respectively the maximum speed and maximum acceleration of the UAV in flight, while $\boldsymbol{l}_{UAV}^{initial}$ and $\boldsymbol{l}_{UAV}^{end}$ respectively represent the initial position and final position of the UAV. η is a non-negative small value.

For the computed energy consumption of the UAVs, it mainly refers to the energy consumed by the UAVs to process the task of SMDs offload. The computed energy consumption $E^t_{com,j}$ of UAV j in time slot t is expressed as follows, that is

$$E^t_{com,j} = k \cdot \left(\sum_{i=1}^{N} f_{i,j} \right) \cdot \tau, \tag{27}$$

where k is the effective switching capacitance of the CPU in the UAV group and is determined by the hardware architecture of the CPU [36].

As for the task migration energy consumption generated by UAV, as the name implies, which is the transmission energy consumption caused by UAV during task migration. The migration energy consumption is mainly determined by the transmission power of UAV. We use $E^t_{mig,j}$ to denote the energy consumption of UAV j migrate tasks to other UAVs or the BS in time slot t, and the specific definition is as follows,

$$E^t_{mig,j} = \begin{cases} p^t_{UAV} \cdot \tau, \text{ if the migration destination is a UAV,} \\ p^t_{j,BS} \cdot \tau, \text{ if the migration destination is a BS.} \end{cases} \tag{28}$$

In summary, the total energy consumption of UAV j in time slot t is expressed as follows, that is,

$$E^t_{total,j} = \phi_1 \cdot E^t_{fly,j} + \phi_2 \cdot E^t_{com,j} + \phi_3 \cdot E^t_{mig,j}, \tag{29}$$

where ϕ_1, ϕ_2, and ϕ_3 are weighting factors, which coordinate the proportion of the three energy consumptions of UAVs' flight energy consumption, UAVs' computing energy consumption and UAVs' migration energy consumption.

3.5. Problem Formulation

In the previous content of the paper, we introduced the UAV-enabled MEC system and established mathematical models from four aspects, that is, architecture, communication, delay, and energy consumption. To sum up, we have three main optimization goals, and as shown below.

- Reduce the energy consumption of UAVs. Since the power of UAVs is limited, it is particularly important to reduce the energy consumption of UAVs. In the previous mathematical model, we have learned that the energy consumption of UAVs is mainly related to the three aspects of flight energy consumption, computing energy consumption, and migration energy consumption. Since the flying range of UAVs is fixed, we mainly extend the stagnation time of UAVs by reducing the computational energy consumption and migration energy consumption, to avoid some UAVs from leaving the UAV group early due to excessive energy consumption.
- Achieve the load balancing of the UAV group. When the MEC system needs to migrate SMD offload tasks, in addition to the migration energy consumption, we also need to pay close attention to the load status of each UAV in the UAV group. If load of a certain UAV is too high, we will not use it as an "endpoint edge server" to provide services to SMDs, so the MEC system will choose a UAV with a lower load to provide services to SMDs. In the end, we can make full use of the computing resources of the UAV group and achieve load balancing for the entire UAV group.
- Improve the service quality of the users. In addition to the above two points, user service quality cannot be ignored. In the process of dynamically migrating tasks offloaded by SMDs, it is inevitable that "service interruption" will occur. We believe that the shorter the service interruption time, the higher the service quality of the users. In this model, we will set a threshold. If the time required for task migration is higher than the threshold, we will not adopt this migration strategy and choose another migration strategy to complete the dynamic migration of tasks for SMDs until

the migration time is less than the threshold, so as to ensure service quality of the users.

In summary, we define the following objective function, and also include the above Equations (2), (6), (10), (13), (16), (17) and (24)–(26).

$$\begin{aligned} \boldsymbol{P1}: \min_{\widetilde{\pi}} & \sum_{t=0}^{K-1} \sum_{j=1}^{M} E_{total,j}^{t} \\ s.t.\ L_j^t \ & \leq \ \frac{\sum_{j=1}^{M} L_j^t}{M} + \epsilon, j \in \mathcal{M}, t \in \mathcal{T}, \\ T_{i,j}^{mig} \ & \leq \ \delta, i \in \mathcal{N}, j \in \mathcal{M}, \end{aligned} \tag{30}$$

where $\widetilde{\pi}$ represents the migration strategy. ϵ represents a minimum error value. The purpose of setting the minimum error value is to improve the robustness in the MEC system. As for δ, it is a threshold used to limit the task migration time. The inequality $L_j^t \leq \frac{\sum_{j=1}^{M} L_j^t}{M} + \epsilon, j \in \mathcal{M}, t \in \mathcal{T}$ and $T_{i,j}^{mig} \leq \delta, i \in \mathcal{N}, j \in \mathcal{M}$ are to ensure that the load balancing of the UAV group and the service quality of the users, respectively.

Obviously, there are many variables in problem ***P1***, and these variables will change with time, making it difficult to find the optimal solution, so we use the Markov decision model to find the optimal task migration strategy.

4. Dynamic Task Migration Strategy

4.1. Markov Decision Processes with Unknown Rewards (MDPUR)

In this section, we will introduce the algorithm of the paper. In the UAV-enabled MEC scenario, the information interaction between the MDs and the UAVs is affected by the environment, such as the wind speed and air density that we mentioned earlier. Besides, the service quality of the users is also the focus of our attention. In summary, how to get the optimal migration strategy under the circumstances of not only paying attention to environmental factors but also considering service quality has become a difficult point.

In this paper, we use the MDPUR algorithm to solve this problem. The MDPUR algorithm is different from the traditional MDP. The MDPUR algorithm is an improvement over MDP. In particular, the MDPUR algorithm reduces the number of value iterations utilizing classification and filtering, thereby increasing the speed of the algorithm. The specific improvements of the MDPUR algorithm are as follows.

- Consider the dominance vector. In the process of solving the optimal migration strategy, we considered the dominance vector. In a given initial state, each action will change the state of the UAVs. And we will classify the result of value iteration. The advantage of this is that it can improve the accuracy of solving the optimal task migration strategy.
- Improve the traditional value iteration method. The previous value iteration solution is too complicated, which will lead to high complexity of the algorithm, so we have improved on the traditional value iteration method. After each value iteration is completed, we will calculate the probability of each vector value function in the dominant vector set as the parent vector, and then select the two groups with the highest probability as the parent and perform the cross operation, and finally calculate the new descendant value function. If the result is better than all the previously calculated value functions, then use it to replace the optimal solution of the current iteration. Otherwise, give up the processing of the offspring. We call the above-mentioned value iteration method "parent crossover algorithm", which can reduce

the total number of MDPUR median iterations and increase the running speed of the MDPUR algorithm in the MEC system.

- Set dynamic UAV status. In the MDPUR algorithm, each UAV is equivalent to a node on the migration path. We define the state of the node as dynamically changing, which is more in line with the UAV-enabled MEC scenario and proves the feasibility of the algorithm. In the end, it is beneficial to apply the algorithm to real life.

In this paper, the MDPUR algorithm finally obtains the optimal task migration strategy after considering the distance of task migration, the residual energy state of the UAVs, and the load state of the UAVs. Not only does the algorithm not occupy too much computing resources of the UAVs, but it also has a very strong ability to find solutions. Especially in a dynamic environment, the MDPUR algorithm has very high flexibility and robustness. In the following content, we directly define a UAV as a node on the task migration path, which is more conducive to understanding. For example, node j represents UAV j.

4.2. Mathematical Model of the MDPUR Algorithm

In this section, we begin to establish the mathematical model on which MDPUR is based. Similar to traditional MDP, we define a six-tuple $\{\mathcal{S}, \mathcal{A}, \widetilde{p}, r, \gamma, \beta\}$, where $\mathcal{S}$ is the set of state space. $\mathcal{A}$ is the set of actions, and $\widetilde{p}$ is the state transition probability. As for $r : \mathcal{S} \times \mathcal{A} \to R$ and $\gamma \in [0, 1]$, they represent the reward function and the discount factor, respectively. The discount factor indicates the importance of future reward relative to current reward. The last item β in the six-tuple is the initial state distribution of the nodes. Next, we conduct a specific analysis of the above.

4.2.1. State Space Set, Action Set, State Transition Probability

Because the state of each UAV changes with time, we use S_j^t to denote the state of UAV j in time slot t. Based on that the system contains M UAVs, we define $\mathcal{S} = \{S_1^t, S_2^t, ..., S_j^t, ..., S_M^t\} \forall j \in \mathcal{M}, \forall t \in \mathcal{T}$ to represent the state space set of the UAVs. Moreover, we consider the three aspects of the UAVs' position, the UAVs' residual energy, and the UAVs' load into the state space of the MEC system. Therefore, S_j^t is a triple and $S_j^t = \{l_{UAV,j}^t, b_j^t, p_j^t\}$. Among them, $l_{UAV,j}^t$ is the position of UAV j in time slot t. b_j^t and p_j^t are the residual energy state and load state of UAV j in time slot t, respectively. Next, we introduce the residual energy state and load state of the UAVs in detail.

For the residual energy state of the UAVs, since most of the energy supply of the UAVs are derived from the lithium battery carried by itself, we define C_j^t to represent the residual energy of UAV j in time slot t, in joules (J). Because we have already defined the total energy consumption of UAV j in time slot t, that is $E_{total,j}^t$, the iteration formula of C_j^t is as follows

$$\begin{cases} C_j^{t+1} = C_j^t - \phi_1 \cdot E_{fly,j}^t - \phi_2 \cdot E_{con,j}^t - \phi_3 \cdot E_{mig,j}^t = C_j^t - E_{total,j}^t, \\ C_j^1 = C_j^{max}, \end{cases} \tag{31}$$

where C_j^1 is the residual energy state of UAV j in the first time slot (that is, the initial energy state), and C_j^{max} is the maximum residual energy of UAV j. To understand the residual energy state of UAV j in time slot t more intuitively, we define b_j^t as a percentage, which can intuitively represent the residual energy as a percentage of its maximum energy. In particular, the specific formula corresponding to b_j^t is as follows.

$$b_j^t = \frac{C_j^t}{C_j^{max}}. \tag{32}$$

For the load status of the UAV. We believe that the UAV task queue length can indirectly reflect the load status of the UAV, that is, the greater the UAV task queue length,

the higher the UAV load. We define p_j^t to represent the load state of UAV j in time slot t, the specific formula is as follows.

$$p_j^t = \frac{L_j^t}{L_{UAV,j}^{max}} \cdot 100\%. \tag{33}$$

Similar to b_j^t, p_j^t is also a percentage, which can visually indicate the load status of the UAV. For example, $p_j^t = 90\%$ indicates that the load of UAV j in time slot t is 90%. At this time, UAV j needs to process a large number of tasks and the load is high.

After building the state space Set of the UAVs, we now start to build the action Set of the UAVs. In the MDPUR algorithm, when the system acts, the state of the corresponding UAV will change accordingly. There are M nodes in the paper, so we can think that the action set $\mathcal{A}$ contains M actions. The specific formula is as follows.

$$\mathcal{A} = \{a_j | j = 1, 2, ..., M\}, \tag{34}$$

where a_j refers to the task is migrated to UAV j

As for the state transition probability, we define $\widetilde{p}(s'|s, a)$ to represent the state transition probability that the state changes from s to s' after performing action a.

4.2.2. Reward Function

In general, the size of the reward function r can directly reflect whether a certain state or action is good or bad for the current state, and we can understand it as a numerical cost. In this paper, we will comprehensively consider the two information of the load state and residual energy state of the UAV. Different from the previous MDP, the value of the reward function r in this paper is obtained by the dot product calculation of two vectors, so the definition of the reward function is as follows.

$$\begin{cases} \widetilde{\lambda} = (\omega_1, \omega_2), \\ \widetilde{r}_{s,a_j} = (1 - \rho_j^t, b_j^t), \end{cases} \tag{35}$$

$$r_{s,a_j} = \widetilde{\lambda} \cdot \widetilde{r}_{s,a_j} = \widetilde{\lambda} * (\widetilde{r}_{s,a_j})^T = \omega_1 \cdot (1 - \rho_j^t) + \omega_2 \cdot b_j^t, \tag{36}$$

where $\widetilde{\lambda}$ and $\widetilde{r}_{s,a_j}$ are binary vectors. r_{s,a_j} is the value of the reward function obtained by selecting action a_j for state s. When the load of the UAV is lower, r_{s,a_j} is larger. Similarly, when the residual energy of UAV is more, r_{s,a_j} is larger; on the contrary, when the UAV load is higher or the residual energy is less, r_{s,a_j} is smaller. In addition, ω_1 and ω_2 are weighting factors, which satisfy $\omega_1 + \omega_2 = 1$, $0 \leq \omega_1 \leq 1$ and $0 \leq \omega_2 \leq 1$. Although the load state of UAV and the residual energy state of UAV are considered by the MDPUR algorithm at the same time, the priority of the two is different in different situations. The following describes the case where the load state or the residual energy state is considered first under the two conditions.

In condition 1, after the UAV group just took off, the UAVs in the UAV group have more residual energy at this time, so when the MEC system performs task migration for the task offloaded by a SMD, we first consider the load state of the UAVs, followed by the residual energy state of the UAVs. Therefore, the value of ω_1 will increase and the corresponding value of ω_2 will decrease. In condition 2, when $\frac{1}{M} \cdot \sum_{j=1}^{M} \leq 50\%$, that is to say, the overall residual energy of the UAV group is low, in order to maximize the lag time of the UAVs with the lower residual energy in the UAV group, we first consider the residual energy state of the UAVs, followed by the load state of the UAVs. Therefore, the value of ω_1 will decrease and the corresponding value of ω_2 will increase. After introducing the reward function in the model, we will define the value function in the next section.

4.2.3. Value Function

Before introducing the value function, we first introduce a concept "long-term expected reward", the value of the long-term reward can directly reflect the advantages and disadvantages of the strategy. In theory, we can use a long-term expected reward to evaluate the pros and cons of any strategy. In other words, the value function is the concrete expression of the long-term expected reward. The value function $V^{\widetilde{\pi}}(s)$ in this paper is different from the value function in the general MDP, the specific definition is as follows.

$$V^{\widetilde{\pi}}(s) = r_{s,\widetilde{\pi}(s)} + \gamma \cdot \sum_{s' \in S} \widetilde{p}(s'|s, \widetilde{\pi}(s)) \cdot V^{\widetilde{\pi}}(s'), \tag{37}$$

where $\widetilde{\pi}$ represents the strategy, it is a mapping from the state set to the action set, namely $\widetilde{\pi} : \mathcal{S} \to \mathcal{A}$. Whereas $\widetilde{\pi}(s)$ refers to taking action $\widetilde{\pi}(s)$ in state s, $V^{\widetilde{\pi}}(s)$ represents the value function of each strategy, that is, $\widetilde{\pi}(s) : \mathcal{S} \to \nabla$. s' is the next state. Formula (37) is the recursive regression form of the value function, from which we can understand that the value of a state is composed of the reward of the state and the subsequent state value in a certain proportion. And We usually call the Formula (37) the Bellman equation. Regarding γ, we have already introduced it earlier. It means that the conversion factor and the value range is $0 \leq \gamma \leq 1$. Especially, The role of the conversion factor is to ensure that the value function converges during the recursive process.

In this model, we also consider the initial states of the UAVs, and the formula is defined as follows.

$$\mathbb{E}_{s,\beta}[V^{\widetilde{\pi}}(s)] = \sum_{s \in \mathcal{S}} \beta \cdot V^{\widetilde{\pi}}(s) = \beta \cdot V^{\widetilde{\pi}} \tag{38}$$

4.3. Markov Decision Processes with Unknown Rewards (MDPUR) Algorithm

Now, we start to solve the optimal migration strategy $\widetilde{\pi}^*$. Among all possible MDP strategies, the expected value of $\widetilde{\pi}^*$ is the largest, and $\widetilde{\pi}^*$ is expressed as follows

$$\widetilde{\pi}^* = arg \max_{\widetilde{\pi}} \beta \cdot V^{\widetilde{\pi}} \tag{39}$$

Earlier we have carefully introduced the Markov decision process(MDP) model under the UAV-enabled MEC system. Below, we specifically explain the solution process of the joint optimization problem (30). To get the optimal strategy, we have proposed the MDPUR algorithm, which is mainly composed of two parts, namely the dominant value iteration algorithm and the parent crossover algorithm. The specific steps are as follows.

Step 1: Initialization. First, we need to collect the information of the nodes (that is, UAVs) and establish a state set $\mathcal{S}$ and an action set $\mathcal{A}$. The research in the paper is to optimize the three aspects of the energy consumption of the UAV group, the load balancing of the UAV group, and the service quality of users. Therefore, this is a joint optimization problem.

Step 2: Calculate the reward function and value function. In the model, the size of the reward function r is mainly determined by the load state of the node and the residual energy state of the node, which helps the UAV group to achieve load balancing as soon as possible and improve the hang time of the UAV group. Then we can get the information of the target node through the location of the SMD, because the service range of the node may overlap, so the target node in this paper can be one or more. After we have determined the target node, we can calculate the value function.

Step 3: Dominant value iteration. Before iterating over the dominant value, we need to set a threshold ζ and then start the first iteration of the value function. If the absolute value of the mathematical expected difference between the value function $V_n^{\widetilde{\pi}}$ of the n-th iteration and the value function of the $(n + 1)$-th iteration is less than ζ (that is, $||\mathbb{E}_{s,\beta}(V_n^{\widetilde{\pi}}) - \mathbb{E}_{s,\beta}(V_{n+1}^{\widetilde{\pi}})|| \leq \zeta$ is satisfied), then stop the iteration and execute step 5 directly; Otherwise, continue execute step 3. After each iteration, we will classify the value

function and determine the dominant value function in this iteration, and then execute step 4.

Step 4: Perform the parent crossover (PC) algorithm. We use the parent crossover algorithm to process the dominant value function and update the calculation result to the new dominant value function set, and then continue to Step 3.

Step 5: Determine the optimal migration strategy $\widetilde{\pi}^*$. When the iteration is completely stopped, we will sort the value function of the last iteration according to the size of absolute value and choose 5 optimal strategies, and then we will calculate the total energy consumption of the corresponding UAV group. The strategy with the lowest total energy consumption is the optimal strategy $\widetilde{\pi}^*$. If the migration time $T_{i,j}^{mig}$ corresponding to the strategy $\widetilde{\pi}^*$ does not satisfy the Formula (32), return to Step 3 to recalculate.

Next, we will separately introduce how to use the advantage-based value iteration algorithm and the parent crossover algorithm to solve the optimal migration strategy and analyze the performance of the MDPUR algorithm.

4.3.1. Advantage-Based Value Iteration (ABVI) Algorithm

In this section, we propose a new algorithm. We call this algorithm an advantage-based value iteration (ABVI) algorithm. Next, we introduce how to use the ABVI algorithm to get the optimal strategy and analyze the performance of the ABVI algorithm.

First, we define ζ as the threshold for value iteration. When $||\mathbb{E}_{s,\beta}(V_n^{\widetilde{\pi}}) - \mathbb{E}_{s,\beta}(V_{n+1}^{\widetilde{\pi}})|| \leq \zeta$ is satisfied, the value iteration is stopped. Before performing the ABVI algorithm, we need to know how to calculate the corresponding strategy through the value function. Here, we define an intermediate variable that is often used in the solution process, namely the action value function $Q^{\widetilde{\pi}} : \mathcal{S} \times \mathcal{A} \to r$, which is described as follows.

$$Q^{\widetilde{\pi}}(s, a_j) = r_{s,a_j} + \gamma \cdot \sum_{s' \in \mathcal{S}} \widetilde{p}(s'|s, a_j) \cdot V^{\widetilde{\pi}}(s'), \tag{40}$$

where $Q^{\widetilde{\pi}}(s, a_j)$ refers to the choice of action a_j in state s, and other states adopt the expected reward of strategy $\widetilde{\pi}$. When the strategy does not display records and only has the value function $V^{\widetilde{\pi}}$, the action value function is recorded as Q. And the strategy $\widetilde{\pi}$ can be calculated by the following formula.

$$\widetilde{\pi}(s) = arg \max_{a_j \in \mathcal{A}} Q(s, a) \tag{41}$$

At the same time, we calculate the difference $d(s, a_j)$ between the action value function $Q^{\widetilde{\pi}}(s, a_j)$ and the value function $V^{\widetilde{\pi}}(s)$, the specific formula is as follows.

$$d(s, a_j) = Q^{\widetilde{\pi}}(s, a_j) - V^{\widetilde{\pi}}(s). \tag{42}$$

In addition, because the paper also considers the distribution of the initial state of the node, the difference between $Q^{\widetilde{\pi}}(s, a_j)$ and $V^{\widetilde{\pi}}(s)$ corresponding to the mathematical expectation $D(s, a_j)$ is as follows.

$$D(s, a_j) = \mathbb{E}_{s,\beta}[Q^{\widetilde{\pi}}(s, a_j)] - \mathbb{E}_{s,\beta}[V^{\widetilde{\pi}}(s)] = \beta\{Q^{\widetilde{\pi}}(s, a_j) - V^{\widetilde{\pi}}(s)\}. \tag{43}$$

Moreover, $d(s, a_j)$ in Formula (42) and $D(s, a_j)$ in Formula (43) can also be called advantages, and these two values will be applied in the later iteration of the advantage value. The so-called advantage-based value iteration is to transform the traditional value function into a vector value function, and then iterate the value. After each iteration, we compare and classify these vector-valued functions. Then, the advantage-based value function set continues to the next round of value iterations until it is satisfied $||\mathbb{E}_{s,\beta}(V_n^{\widetilde{\pi}}) - \mathbb{E}_{s,\beta}(V_{n+1}^{\widetilde{\pi}})|| \leq \zeta$ stops iterating.

In the process of advantage-based value iteration, it is undoubtedly very difficult to compare vector-valued functions. Here, we define two methods for comparing vectors. The first vector comparison method is pareto dominance, which is a very common vector comparison method and described in Definition 1.

Definition 1. *For a given two* d*-dimensional vectors* $\lambda_1 = (q_1, q_2, ..., q_d)$ *and* $\lambda_2 = (p_1, p_2, ..., p_d)$*, we can know*

$$\lambda_1 \succ \lambda_2 \Leftrightarrow q_i \geq p_i, \forall i, 0 \leq i \leq d. \tag{44}$$

when $q_i \geq p_i$ *is satisfied, vector* λ_1 *is better than vector* λ_2*.*

The second vector comparison method is the vector dot product. In the previous content, we defined $\widetilde{\lambda}$. For a given two two-dimensional vectors, $\lambda_1 = (q_1, q_2)$ and $\lambda_2 = (p_1, p_2)$, we have

$$\widetilde{\lambda} * (\lambda_1)^T \geq \widetilde{\lambda} * (\lambda_2)^T \Rightarrow \lambda_1 \succ \lambda_2. \tag{45}$$

Here, we will calculate the dot $\widetilde{\lambda}$ with λ_1 and λ_2, respectively. Because they are two-dimensional vectors, the calculation result is a real number, and then we compare the vectors by comparing the size of the two real numbers.

Next, we begin to define the vector value function, because the reward function of this paper is mainly determined by the load state and residual energy state of the nodes, so the corresponding vector value function is two-dimensional, the specific formula is as follows.

$$\overline{V}^{\widetilde{\pi}}(s) = \overline{r}_{s,a_j} + \gamma \cdot \sum_{s' \in \mathcal{S}} \widetilde{p}(s'|s, \widetilde{\pi}(s)) \cdot \overline{V}^{\widetilde{\pi}}(s'), \forall s, \tag{46}$$

$$\overline{r}_{s,\widetilde{\pi}(s)} = (\omega_1 * (1 - \rho_j^t), \omega_2 * b_j^t). \tag{47}$$

Among them, $\overline{r}_{s,\widetilde{\pi}(s)}$ is a two-dimensional vector, and it is calculated from vector $\widetilde{\lambda}$ and vector $\widetilde{r}_{s,a_j}$. As for the calculation method, it is a new vector formed by multiplying the corresponding elements in the vector.

After determining the vector value function, we can easily get the mathematical expectation of the vector value function considering the initial state distribution of the node, and it is expressed as follows.

$$\overline{V}^{\widetilde{\pi}} = \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}}(s)] = \sum_{s \in \mathcal{S}} \cdot \overline{V}^{\widetilde{\pi}}(s). \tag{48}$$

In the process of vector value iteration, we also need to define $\widetilde{\pi}^{\hat{s},\hat{a}}$ to solve the set of advantage-based value function, and $\widetilde{\pi}^{\hat{s},\hat{a}}$ is expressed as follows.

$$\widetilde{\pi}^{\hat{s},\hat{a}} = \begin{cases} \widetilde{\pi}(s), \ if \ s \neq \hat{s}, \\ \hat{a}, \ if \ s = \hat{s}. \end{cases} \tag{49}$$

Definition 2. *If* $\overline{V}^{\pi}$ *is not the optimal vector value function, then there must be a state* $\bar{s}$ *that makes* $\widetilde{\lambda} * (\overline{V}^{\widetilde{\pi}^{\hat{s},\hat{a}}})^T \geq \widetilde{\lambda} * (\overline{V}^{\widetilde{\pi}})^T$ *true after selecting action* $\bar{a}$*.*

Because the only difference between $\widetilde{\pi}$ and $\widetilde{\pi}^{\hat{s},\hat{a}}$ is state $\hat{s}$, the vector action value function satisfies the following inequality.

$$\widetilde{\lambda} \cdot \widetilde{Q}^{\widetilde{\pi}}(\hat{s}, \widetilde{\pi}^{\hat{s},\hat{a}}(\hat{s})) = \widetilde{\lambda} \cdot \overline{V}^{\widetilde{\pi}^{\hat{s},\hat{a}}}(\hat{s}) \geq \widetilde{\lambda} \cdot \overline{V}^{\widetilde{\pi}}(\hat{s}) \tag{50}$$

Next, we take $\widetilde{\pi}^{\hat{s},\hat{a}}$ and $\widetilde{\pi}$ as examples to analyze the advantages of $\widetilde{\pi}^{\hat{s},\hat{a}}$. Therefore, Equation (50) can be obtained.

$$\overline{D}(\hat{s},\hat{a}) = \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}^{\hat{s},\hat{a}}}] - \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}}] = \sum_{s\in\mathcal{S}} \beta \cdot \overline{V}^{\widetilde{\pi}^{\hat{s},\hat{a}}}(s) - \sum_{s\in\mathcal{S}} \beta \cdot \overline{V}^{\widetilde{\pi}}(s) \tag{51}$$

In Formula (51), $\overline{D}(\hat{s},\hat{a})$ is the difference between the mathematical expectation of the vector value function $\widetilde{\pi}^{\hat{s},\hat{a}}$ and the mathematical expectation of the vector value function $\widetilde{\pi}$, which is also called the advantage of the vector. Because the difference between $\widetilde{\pi}^{\hat{s},\hat{a}}$ and $\widetilde{\pi}$ is the state $\hat{s}$, we can finally get the advantage of $(\hat{s},\hat{a})$ by combining Formulas (50) and (51), that is,

$$\begin{aligned} \overline{D}(\hat{s},\hat{a}) &= \beta \cdot \overline{V}^{\widetilde{\pi}^{\hat{s},\hat{a}}}(\hat{s}) - \beta \cdot \overline{V}^{\widetilde{\pi}}(\hat{s}) \\ &= \beta\{\bar{Q}^{\widetilde{\pi}}(\hat{s},\widetilde{\pi}^{\hat{s},\hat{a}}(\hat{s})) - \overline{V}^{\widetilde{\pi}(\hat{s})}\} \\ &= \beta\{\overline{Q}^{\widetilde{\pi}}(\hat{s},\hat{a}) - \overline{V}^{\widetilde{\pi}(\hat{s})}\}. \end{aligned} \tag{52}$$

The above is the process of the first round of vector value function iteration. In the subsequent iterations, we assume that $(\hat{s}_2,\hat{a}_2)$, $(\hat{s}_3,\hat{a}_3)$, or $(\hat{s}_n,\hat{a}_n)$ may appear in addition to $(\hat{s},\hat{a})$. Therefore, we define $\widetilde{\pi}'' = \widetilde{\pi}^{\hat{s}_2,\hat{a}_2}$ to indicate that the state $hats_2$ selects the action $hata_2$ in the second round of iteration. Moreover, the role of $\widetilde{\pi}''$ is similar to the previous $\widetilde{\pi}^{\hat{s},\hat{a}}$. Hence, we have inequality (53) in the second round of value iteration.

$$\lambda \cdot \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}''}] \geq \lambda \cdot \overline{V}^{\widetilde{\pi}^{\hat{s}_1,\hat{a}_1}} \geq \lambda \cdot \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}}]. \tag{53}$$

Correspondingly, the advantages of the vector value function in the second round are as follows.

$$\begin{aligned} \overline{D}(\hat{s}_2,\hat{a}_2) &= \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}''}] - \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}}] \\ &= \mathbb{E}_{s,\beta}[\overline{Q}^{\widetilde{\pi}}(s,\widetilde{\pi}''(s))] - \mathbb{E}_{s,\beta}[\overline{V}^{\widetilde{\pi}}] \\ &= \sum_{s\in\mathcal{S}} \beta\{\bar{Q}^{\widetilde{\pi}}(\hat{s}_2,\hat{a}_2) - \overline{V}^{\widetilde{\pi}}(s)\}. \end{aligned} \tag{54}$$

To speed up the iteration and try to reduce the complexity of the algorithm, we will use the parent crossover algorithm to process the set of advantage vectors after each iteration. And the parent crossover algorithm will be introduced in detail in the next section. Moreover, the ABVI algorithm also considers the initial distribution of the nodes. The pseudo code of the ABVI algorithm is described in Algorithm 1. In Algorithm 1, $\overline{D}_{adv}$ is the set of advantage vectors, and n is the number of iterations.

Algorithm 1 The proposed ABVI algorithm

Input: $\{\mathcal{S}, \mathcal{A}, \widetilde{p}, r, \gamma, \beta\}, \zeta$
Output: $\widetilde{\pi}^*$

$t \leftarrow 0, n \leftarrow 1$;
$\widetilde{\pi}^* \leftarrow$ define a strategy as the optimal strategy randomly;
/* set all vector value functions to 0 vectors before the iteration starts */
$\overline{V}_0(s) = (0,0), \forall s \in \mathcal{S}$;
Set the ω_1 and ω_2 in the reward function according to the overall energy state of the UAV group;
repeat
    $\overline{D}_{adv} \leftarrow \Phi$
    for each $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
        $Q_n^{\widetilde{\pi}}(s, a_j) \leftarrow \bar{r}_{s,a_j} + \gamma \cdot \sum_{s' \in \mathcal{S}} \widetilde{p}(s'|s,a) \cdot \overline{V}_n^{\widetilde{\pi}}(s')$;
        $\overline{D}(s,a) = \beta\{\overline{Q}^{\widetilde{\pi}}(s,a) - \overline{V}^{\widetilde{\pi}(s)}\}$;
        Add $\overline{D}(s,a)$ to $\overline{D}_{adv}$;
    end for
    Enter $\overline{D}_{adv}$ into the Algorithm 2;
    Receive the result of the Algorithm 2, that is, $\overline{D}_{adv}^{new}$;
    for each $s \in \mathcal{S}$ **do**
        $\overline{V}^{\widetilde{\pi}}(s) = \bar{r}_{s,\widetilde{\pi}(s)} + \gamma \cdot \sum_{s' \in \mathcal{S}} \widetilde{p}(s'|s, \widetilde{\pi}(s)) \cdot \overline{V}^{\widetilde{\pi}}(s')$;
        Find $\beta \cdot \|\overline{V}_{n+1}^{\widetilde{\pi}}\|$;
        Update $\widetilde{\pi}^* = \widetilde{\pi}$;
    end for
    $n \leftarrow n + 1$;
until $\|\mathbb{E}_{s,\beta}(V_n^{\widetilde{\pi}}) - \mathbb{E}_{s,\beta}(V_{n+1}^{\widetilde{\pi}})\| \leq \zeta$

4.3.2. Parent Crossover (PC) Algorithm

In this section, we introduce the parent crossover algorithm that is, Algorithm 2 in the paper.

The idea of the algorithm comes from the ant colony optimization (ACO) algorithm. The algorithm performs a cross operation on a better solution, thereby increasing the possibility of seeking the optimal solution. In Algorithm 1, a new set of advantage vectors will be generated after each round of iteration, and the role of Algorithm 2 is to process these advantage vectors. For Algorithm 2, more specifically, we first need to calculate the probabilities of these vectors as the parent and then select two vectors with the highest probability as the parent vector for the cross operation. Finally, we get the new vector after the cross operation. If the new vector is better than the optimal vector value function in the advantage vector set, the new vector is added to the advantage vector set; otherwise, the advantage vector set is not processed. The parent crossover algorithm is shown in Algorithm 2 in the form of pseudo code.

Algorithm 2 The proposed PC algorithm

Input: $\bar{D}_{adv}$
Output: $\bar{D}_{adv}^{new}$

1: Initialize $\overline{D}_{adv}$ so that $\bar{D}_{adv}$ contains K vectors;
2: **for** each $1 \leq \widetilde{k} \leq K$ **do**
3: Calculate the probability of being the parent of the $\widetilde{k} - th$ vector value function by $p(\widetilde{k}) = \frac{\frac{1}{\|\overline{V}_{\widetilde{k}}^{\widetilde{\pi}}\|}}{\sum_{i=1}^{K} \frac{1}{\|\overline{V}_{i}^{\widetilde{\pi}}\|}} = \frac{1}{\overline{V}_{\widetilde{k}}^{\widetilde{\pi}} \cdot \sum_{i=1}^{K} \frac{1}{\overline{V}_{i}^{\widetilde{\pi}}}}$;
4: **end for**
5: Find two vectors with the highest probability as parents, and define these two vectors as vector $\overline{u}_1$ and vector $\overline{u}_2$;
6: Perform a cross operation on the two parents obtained in the previous step, that is, $\bar{u}^{new} = \overline{u}_1 \times \overline{u}_2$;
7: Compare the optimal value function vector $\overline{V}_{best}^{\widetilde{\pi}}$ in vector $\overline{u}^{new}$ and vector $\overline{D}_{adv}$;
8: **if** $\overline{u}^{new} \succ \overline{V}_{best}^{\widetilde{\pi}}$ **then**
9: $\overline{D}_{adv}^{new} \leftarrow$ Add $\overline{u}^{new}$ to $\overline{D}_{adv}$;
10: **else**
11: $\overline{D}_{adv}^{new} \leftarrow \overline{D}_{adv}$
12: **end if**
13: Update $\overline{D}_{adv}^{new}$ and return it to Algorithm 1;

5. Experimental Results and Discussion

In this section, we use Matlab software to do simulation experiments and evaluate the performance of the ABVI algorithm based on the experimental results. Finally, we proved the efficiency and feasibility of the ABVI algorithm through experiments. Next, we will describe the four aspects of the total energy consumption of the UAV group, the task migration time, the load state of the UAV group, and the flight time of the UAV group.

5.1. Setting of Experimental Parameters

In the simulation experiment of this paper, we set a flying area of a UAV group, the area is a rectangle of 1000 m ×1000 m. The UAV group provides services for SMD in this area. The horizontal movement range of each UAV is 550 square meters. Each UAV has a fixed flight trajectory during flight and the respective flight trajectories do not overlap. In the paper, the UAV-enabled MEC system includes both a dynamic edge server (that is, the UAV group) and a static edge server (that is, a BS). The maximum service range of each UAV in the UAV group is a circular area with a radius of 170 m, and the service range of UAV will also change with the change of UAV height. In addition, other relevant experimental parameters in this paper are defined in Table 2.

Table 2. Definition of the main experimental parameters.

Parameters	Value
The number of UAV	$M = 35$
The number of SMD	$N = 200$
The bandwidth of the network	$B = 10$ MHz
The angle between UAVs' directional antenna and vertical	$\theta = 30^{\circ}$
The initial height of the UAVs	50 m
The maximum flying height of UAVs	$H_{max} = 1000$ m
The maximum transmission power of SMDs	$p_{SMD}^{max} = 2.1$ W
The maximum transmission power of UAVs	$p_{UAV}^{max} = 150$ W
The noise power of SMDs	$\sigma_{SMD}^2 = 1.69 \times 10^{-9}$ W
The noise power of UAVs	$\sigma_{UAV}^2 = 2.3 \times 10^{-8}$ W
The maximum CPU cycle frequency of UAVs	$f_{UAV}^{max} = 25$ GHz
The maximum memory of UAVs	10 T
The density of UAV processing tasks	$\widetilde{c} = 10^3$ cycles/bit
The UAV weight	$\widetilde{M}_j = 20.9$ kg
The air density	$\widetilde{\rho} = 1.313$ kg/m^3
The range of wind speed	[2.5 m/s, 7.1 m/s]
The acceleration of gravity	g = 9.8 m/s^2
The UAV propeller rotations per second	$50 \leq c_j \leq 160$
The maximum speed of UAV	$v^{max} = 13$ m/s
The maximum acceleration of UAV	$a^{max} = 7.5$ m/s^2
The CPU effective capacitance switch of UAV	$\widetilde{k} = 3 \times 10^{-16}$

In the ABVI algorithm of the paper, we also set $\varsigma = 10^{-4}$ and $\gamma = 0.95$. In the following experiment, in order to be more convincing, we also introduced three algorithms of the traditional value iteration (TVI), ant colony optimization(ACO), and distance-based MDP (MDP-SD) to compare with the proposed ABVI algorithm. For ACO algorithm, when ants are looking for food sources, they can release a hormone called pheromone on the path they travel, so that other ants within a certain range can detect it. When there are more and more ants passing through some paths, there will be more and more pheromone, and the probability of ants choosing this path will be higher. As a result, the pheromone on this path will increase again, and the ants will follow this path. The probability of road increases again.

5.2. Analysis of the Total Energy Consumption of the UAV Group

Now, we analyze the total energy consumption of the UAV group. Because the energy of UAVs is limited, it is very necessary to reduce the energy consumption of UAVs. In the previous content, we have introduced the energy consumption model of UAVs, and we found that reducing the cost of task migration is very effective for reducing the total energy consumption of the UAV group. Therefore, we can know that the quality of the migration strategy will directly affect the total energy consumption of the UAV group. The specific experimental results are shown in Figure 4.

In this experiment, we assume that there are 200 SMDs in the experiment to offload tasks to the UAV group, and the actual number of tasks that need to be migrated after reaching the UAV group conforms to the Poisson distribution. The ordinate of Figure 4 represents the total energy consumption of the UAV group, and the unit is kilojoule. From Figure 4, we can find that the total energy consumption of the UAV group increases with time. Between 0 and 19 min, there are relatively few tasks that need to be migrated for tasks offloaded by SMDs, so the total energy consumption of the UAV group is growing at a slower rate. Between 20 and 35 min, the total energy consumption of the UAV group has grown very fast. Between 36 and 79 min, the growth rate of the total energy consumption of the UAV group began to slow down, because the number of tasks that require task migration begins to decrease. From the experiment, we can find that the curves corresponding to the three algorithms are not smooth. That is because during the

flight of the UAVs, it will be affected by environmental factors such as air density and wind speed, so the total energy consumption of the UAV group will be some irregular changes. In addition, we can also find that the total energy consumption of the UAV group corresponding to the ABVI algorithm is the lowest, the total energy consumption of the UAV group corresponding to the MDP-SD algorithm is second lowest. However, the total energy consumption of the UAV group corresponding to the TVI algorithm is the highest. The main reason for this result is that both the ABVI algorithm and the MDP-SD take distance as an influencing factor into the model, so the migration path is generally a smaller distance, and the corresponding migration costs will also be lower. In contrast, the TVI algorithm does not consider the effect of distance on migration costs.

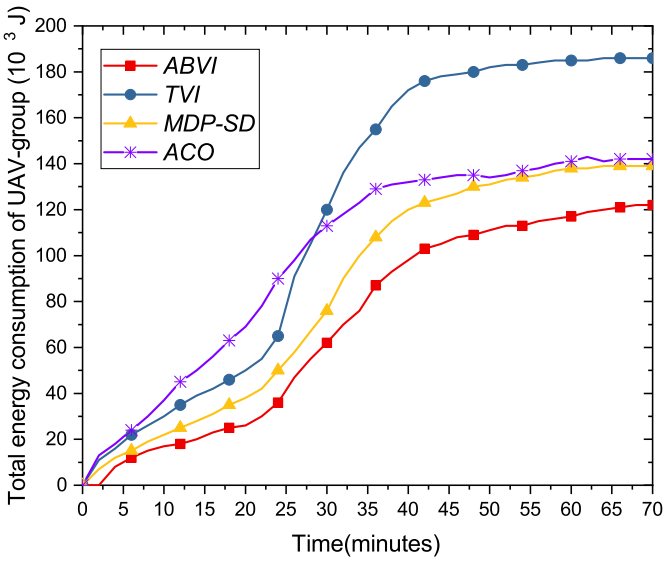

Figure 4. The total energy consumption of the UAV group versus time.

In summary, the proposed ABVI algorithm is effective for reducing the total energy consumption of the UAV group.

5.3. Analysis of Task Migration Time

In this section, we analyze whether the ABVI algorithm can shorten the task migration time to improve the service quality of the users through experiments. As we have mentioned earlier, in the MEC system, shortening task migration time is very important to improve the service quality of the users. The specific experimental results are shown in Figure 5, and the experimental parameters we set are the same as in Figure 4.

In Figure 5, the abscissa represents time in units of minutes, and the ordinate represents average task migration time in seconds. ACO is ant colony optimization. ABVI is the proposed in this paper, TVI and MDP-SD represent the traditional value iteration algorithm and the distance-based MDP algorithm, respectively. Here, the average task migration time represents the average of the time it takes for all SMDs currently in need of task migration to complete the task migration. This average value can very intuitively reflect the service quality of the users. In addition, we also set a standard in the experiment, which is the baseline in Figure 5. As users have higher and higher requirements for real-time services, reducing the time required for task migration as much as possible has become our research focus. Through the experiments, we found that when the service interruption time is less than 0.7 s, the user experience will not be significantly affected. Therefore, we set the baseline to 0.7 s in the experiment. Since we assume that the number of task migrations conforms to the Poisson distribution, the number of task migrations is less between 0 and 19 min, and the average task migration time rises more slowly. However, the number of task migration begins to grow rapidly between 20 min and 35 min, so the

average task migration time will also increase rapidly. As time goes on, the number of task migration begins to slowly decrease between 36 min and 70 min, and the average task migration time gradually decreases and eventually stabilizes. We can also find that the curve corresponding to the ABVI algorithm is always lower than the curves of the other three algorithms. The main reason for this result is that we have fully considered the load status of the UAV group when designing the ABVI algorithm. In contrast, the other two algorithms consider less in this respect, and the complexity of the ABVI algorithm is also lower than the other two algorithms.

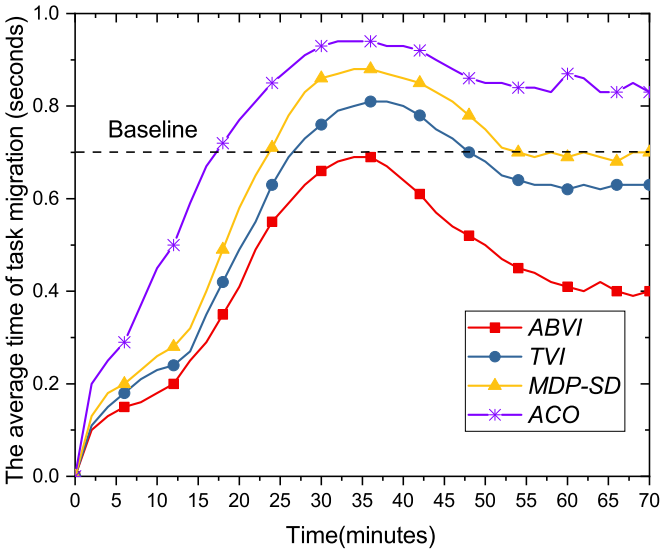

Figure 5. The average task migration time versus time.

In summary, the ABVI algorithm is very effective for reducing task migration time and improving the service quality of the users.

5.4. Analysis of the Load Status of the UAV Group

In this section, we analyze the influence of the ABVI algorithm on the load status of the UAV group through experiments. Here, we analyze from two perspectives, one of which is to calculate the load status of each UAV in the UAV group, and the other is to calculate the difference between the highest load and the lowest load in the UAV group, that is range. The specific experimental results are shown in Figures 6–8.

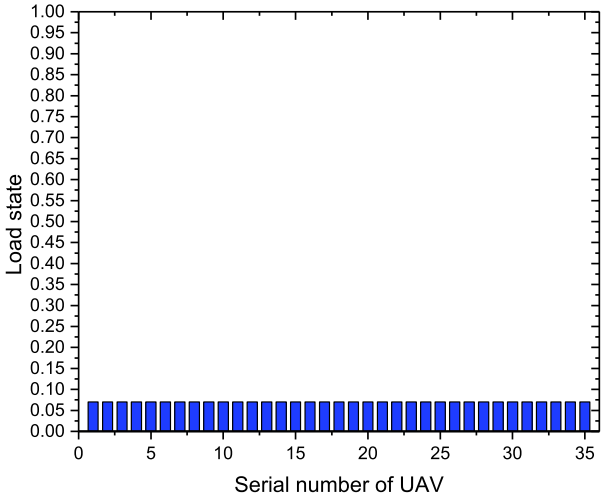

Figure 6. The initial load state of the UAV group.

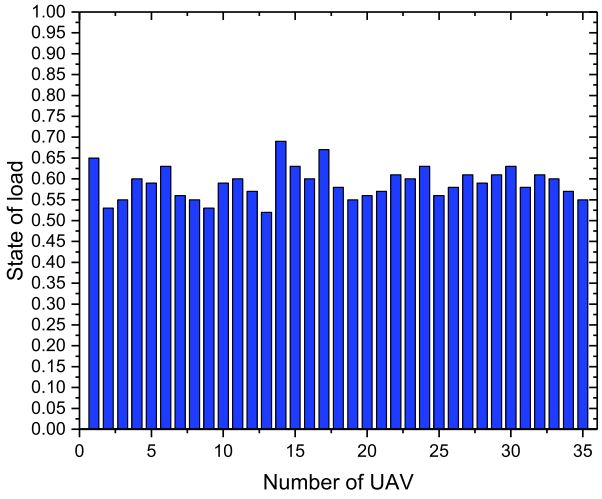

Figure 7. The load status of the UAV group when the UAV-enabled MEC system is running for 35 min.

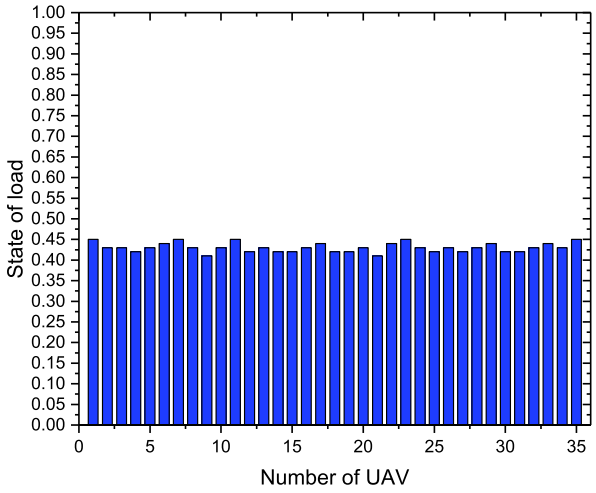

Figure 8. The load status of the UAV group when the UAV-enabled MEC system is running for 65 min.

Figures 6–8 respectively show the load state of the UAV at different times. Here, the abscissa represents the number of each UAV (i.e., $UAV_j, j = 1, 2, ..., 35$), and the ordinate represents the load status. Moreover, we use decimals to represent the load state. For example, 0.7 means that the load state of UAV is 70%, and 1 means that the UAV is at the maximum load state. When the MEC system is just running, each UAV needs to execute an operating system and other programs in the actual scenario, we set the initial load state of each UAV in the UAV group to be 0.07. Figure 6 shows the initial load state of the UAV group. Similar to the experimental parameters in the previous subsection, the UAV group receives the most tasks in 35 min, and it is also the most prone to an unbalanced load of the UAV group at this time. From Figure 7, we can find that although some of the UAVs in the UAV group have higher loads and some have lower loads, the overall trend is balanced and there is no load imbalance. At 65 min, the number of receiving tasks of the UAV group began to decrease. From Figure 8, we can find that the UAV group is still in a state of load balancing.

In summary, by counting the load status of each UAV in the UAV group at different times, we can find that the ABVI algorithm helps the UAV group achieve load balancing during the task migration process and avoiding the unbalanced loads in the UAV group.

Besides, we analyze the load status of the UAV group by calculating the difference between the highest load and the lowest load in the UAV group (i.e., range). For example,

when the highest load of the UAV group is 0.9 and the lowest load is 0.2, then the difference between the two is 0.7, which is the range. The larger the range value, the more unbalanced the load state of the UAV group; otherwise, the smaller the range value, the more balanced the load of the UAV group. The specific experimental results are shown in Figure 9.

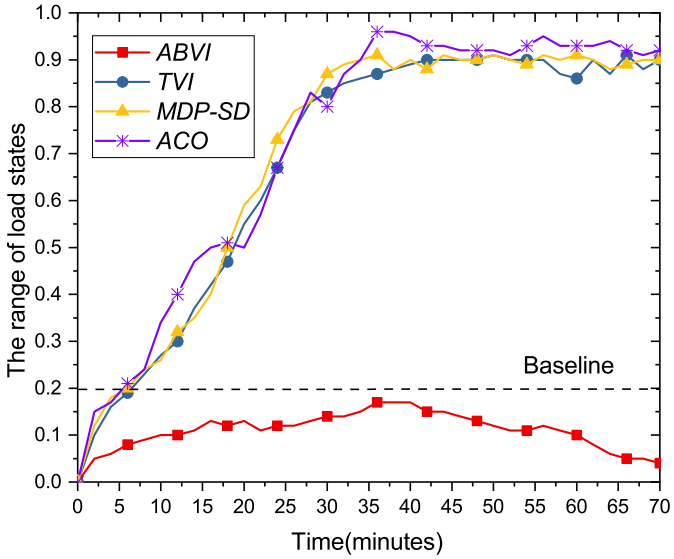

Figure 9. The comparison of the ABVI algorithm and the other two algorithms in optimizing the load of the UAV group.

In the experiment, we set a range baseline of load states, the corresponding value of this baseline is 0.2. More specifically, when the range is less than 0.2, we think that the UAV group is in a state of load balancing. Otherwise, it is in a state of an unbalanced load. From Figure 9, we can find that the curve corresponding to the ABVI algorithm is always lower than the curve corresponding to the other two algorithms. Between 0 and 5 min, the range becomes larger. This is because the number of tasks received by the UAV groups at the beginning is relatively small, so some UAVs are vacant, and the range becomes larger. And between 6 and 70 min, the range sometimes increases slowly but also decreases. The main reason is that the ABVI algorithm considers the load status of the UAV to the reward function, so the ABVI algorithm can dynamically adjust the parameters in the algorithm and make the UAV group achieve load balancing. On the contrary, the remaining two algorithms do not take the UAV load into account, so the curve corresponding to the TVI algorithm and the MDP-SD algorithm has been increasing.

In summary, the ABVI algorithm can help the UAV group achieve load balancing, and the performance of the ABVI algorithm in optimizing the load of the UAV group is higher than the TVI algorithm and the MDP-SD algorithm.

5.5. Analysis of the Flight Time of the UAV Group

In this section, we analyze whether the ABVI algorithm can improve the flight time of the UAV group. The flight time of the UAV group is mainly determined by the residual energy of each UAV in the UAV group. In real life, when the residual energy state of a UAV is below 0.1 (that is, 10%), the UAV must exit the UAV group and land directly to the designated location. Therefore, we consider the residual energy state of the UAV as an influencing factor in the ABVI algorithm, which can avoid that some UAVs consume too much energy due to too many processing tasks and prevent UAVs withdrew from the UAV group prematurely.

In Figure 10, the ordinate represents the range of the residual energy states of the UAV group. This means that we use a range to determine whether the residual energy of the UAV group is in equilibrium. The large range means that the residual energy of

some UAVs in the UAV group is too low and the residual energy of some UAVs is too high, which will cause some UAVs to consume too much energy and exit the UAV group early. However, the reduction in the number of UAVs in the UAV groups will affect the service range and service quality of the users. In this experiment, we set a baseline range of residual energy states, the corresponding value of the baseline is 0.25. When the range of residual energy states is greater than 0.25, we think that the residual energy gap of some UAVs in the UAV group is too large, which will reduce the flight time of the UAV group. On the contrary, When the range of residual energy states is less than 0.25, the residual energy gap of each UAV in the UAV group is small, and the flight time of the UAV group will also increase. From Figure 10, we can find that the curve corresponding to the ABVI algorithm has always been below the baseline.

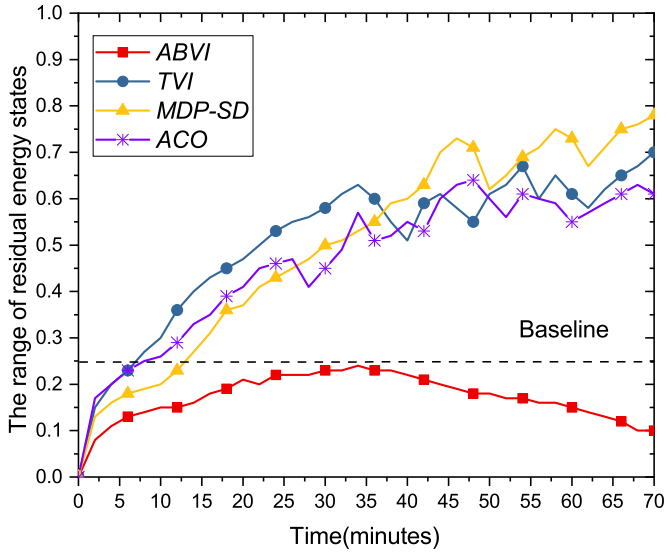

Figure 10. The range of residual energy state versus time.

Besides, we also found that the three curves in the Figure 10 are rising from 0 to 35 min. This is because the number of task migrations needs to be following the Poisson distribution. During this time, the number of tasks that the UAV group needs to process has been increasing, so the range has grown. Between 35 min and 70 min, the curve corresponding to the TVI algorithm and the curve corresponding to the MDP-SD began to fluctuate.There are two reasons for this situation. First, because the TVI algorithm and the MDP-SD algorithm both use distance as a priority condition when choosing target migration nodes, when the number of task migrations begins to slowly decrease, some UAVs that are closer to SMDs still consume high energy, some UAVs farther away from SMDs have lower energy consumption, which causes the range of residual energy state to start to fluctuate. Second, the TVI algorithm and the MDP-SD do not fully consider the impact of the residual energy state of the UAV group on the migration strategy. As for the ABVI algorithm, it can help the residual energy of the UAV group to be in a balanced state. In Formula (38), we take the load state and the residual energy state of the UAV group into the reward function of this model, where ω_1 and ω_2 will be dynamically adjusted according to the average residual energy state of the UAV group, to ensure that the residual energy of the UAV group is in a balanced state and thereby improve the flight time of the UAV group. Besides, the curve corresponding to the ABVI algorithm is not smooth. The main reason is that ω_1 and ω_2 change dynamically during the value iteration process, so the residual energy state of each UAV group changes irregularly.

In summary, the ABVI algorithm can improve the flight time of the UAV group, and ensure the integrity of the UAV group to the greatest extent.

6. Conclusions

In this paper, we combine the dynamic edge servers (that is, the UAVs) and the static edge servers (that is, the BSs) to innovatively establish a three-layer MEC system. The purpose of this paper is to design an effective task migration strategy, which can reduce the total energy consumption of the UAV group as much as possible while ensuring the user service quality and the load balance of the UAV group. Aiming at this joint optimization problem, we established the MDPUR model and proposed the ABVI algorithm combined with the PC algorithm. Moreover, the ABVI algorithm fully considers the influence of the three factors of the load state of the UAVs, the residual energy state of the UAVs, and the migration cost on the MEC system. Finally, the simulation experiments in this paper prove that the ABVI algorithm is more efficient than the traditional TVI algorithm and the MDP-SD algorithm. Through the ABVI algorithm, we can obtain a more reasonable and effective task migration strategy, which can not only ensure the user service quality but also make the UAV group reach a load-balanced state.

Author Contributions: W.O., Z.C., J.W., G.Y. and H.Z. conceived the idea of the paper. W.O., Z.C., J.W. and G.Y. designed and performed the simulations; W.O., Z.C. and J.W. analyzed the data; Z.C. contributed reagents/materials/analysis tools; W.O. wrote the original draft; J.W. and G.Y. reviewed the paper; W.O., Z.C., J.W., G.Y. and H.Z. reviewed the revised papers. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The National Natural Science Foundation of China[61672540]; Hunan Provincial Natural Science Foundation of China [2018JJ3299, 2018JJ3682]; Fundamental Research Funds for the Central University of Central South University (2020zzts597).

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

Abbreviations

The following abbreviations are used in this manuscript:

MEC	mobile edge computing
SMD	smart mobile device
UAV	unmanned aerial vehicle
BS	base station
MDP	Markov decision process
MDPUR	Markov decision process with unknown rewards
ABVI	advantage-based value iteration
PC	parent-generation crossover
ACO	ant colony optimization
TVI	traditional value iteration
MDP-SD	distance-based MDP

References

1. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutorials.* **2017**, *99*, 2322–2358. [CrossRef]
2. Tran, T.X.; Hajisami, A.; Pandey, P.; Pompili, D. Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 54–61. [CrossRef]
3. Wang, S.; Urgaonkar, R.; Zafer, M.; He, T.; Chan, K.; Leung, K.K. Dynamic Service Migration in Mobile Edge Computing Based on Markov Decision Process. *IEEE ACM Trans. Netw.* **2019**, *27*, 1272–1288. [CrossRef]
4. Wang, S.; Zhao, Y.; Huang, L.; Xu, J.; Hsu, C. QoS prediction for service recommendations in mobile edge computing. *J. Parallel Distrib. Comput.* **2019**, *127*, 134–144. [CrossRef]
5. Lim, J.B.; Lee, D.W. A Load Balancing Algorithm for Mobile Devices in Edge Cloud Computing Environments. *Electronics* **2020**, *9*, 686. [CrossRef]
6. Farhan, L.; Shukur, S.T.; Alissa, A.E.; Alrweg, M.; Raza, U.; Kharel, R. A survey on the challenges and opportunities of the Internet of Things (IoT). In Proceedings of the 2017 Eleventh International Conference on Sensing Technology (ICST), IEEE, Sydney, NSW, Australia, 4–6 December 2017.
7. Zeng, Y.; Zhang, R. Energy-Efficient UAV Communication With Trajectory Optimization. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 3747–3760. [CrossRef]

8. Zhan, P.; Yu, K.; Swindlehurst, A.L. Wireless Relay Communications with Unmanned Aerial Vehicles: Performance and Optimization. *Aerosp. Electron. Syst. IEEE Trans.* **2011**, *47*, 2068–2085. [CrossRef]
9. Huang, S.; Lv, B.; Wang, R. MDP-Based Scheduling Design for Mobile-Edge Computing Systems with Random User Arrival. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 Dececember 2019.
10. Ridhawi, I.A.; Aloqaily, M.; Kotb, Y.; Ridhawi, Y.A.; Jararweh, Y. A collaborative mobile edge computing and user solution for service composition in 5G systems. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3446. [CrossRef]
11. Ksentini, A.; Taleb, T.; Chen, M. A Markov Decision Process-based Service Migration Procedure for Follow Me Cloud. In Proceedings of the IEEE International Conference on Communications, IEEE, Sydney, Australia, 10–14 June 2014; pp. 1350–1354.
12. Wang, S.; Urgaonkar, R.; He, T.; Zafer, M.; Chan, K.S.; Leung, K.K. Mobility-Induced Service Migration in Mobile Micro-Clouds. In Proceedings of the 2014 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 6–8 October 2014.
13. Wang, S.; Urgaonkar, R.; He, T.; Zafer, M.; Chan, K.; Leung, K.K. Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), Baltimore, MD, USA, 6–8 October 2014;
14. Ceselli, A.; Premoli, M.; Secci, S. Mobile Edge Cloud Network Design Optimization. *IEEE ACM Trans. Netw.* **2017**, 1818–1831. [CrossRef]
15. Liu, J.; Mao, Y.; Zhang, J.; Letaief, K.B. Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), IEEE, Barcelona, Spain, 10–15 July 2016.
16. Liu, C.F.; Bennis, M.; Poor, H.V. Latency and Reliability-Aware Task Offloading and Resource Allocation for Mobile Edge Computing. In Proceedings of the 2017 IEEE Globecom Workshops, Singapore, 4–8 December 2017.
17. Liu, L.; Chang, Z.; Guo, X.; Ristaniemi, T. Multi-objective optimization for computation offloading in mobile-edge computing. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 832–837.
18. Ahmed, A.; Ahmed, E. A Survey on Mobile Edge Computing. In Proceedings of the International Conference on Intelligent Systems & Control, Coimbatore, India, 7–8 January 2016 .
19. Chen, G. Mobile Research: Benefits, Applications, and Outlooks. In Proceedings of the Internationalization, Design & Global Development-International Conference, Idgd, Held As. DBLP, Orlando, FL, USA, 9–14 July 2011.
20. Zhang, W.; Chen, J.; Zhang, Y.; Raychaudhuri, D. Towards Efficient Edge Cloud Augmentation for Virtual Reality MMOGs. In Proceedings of the The Second ACM/IEEE Symposium on Edge Computing (SEC 2017), San Jose, CA, 12–14 October 2017.
21. Chen, L.; Li, H. An MDP-based vertical handoff decision algorithm for heterogeneous wireless networks. In Proceedings of the Wireless Communications & Networking Conference, Doha, Qatar, 3–6 April 2016.
22. Xie, S.; Chu, X.; Zheng, M.; Liu, C. A composite learning method for multi-ship collision avoidance based on reinforcement learning and inverse control. *Neurocomputing* **2020**, *411*. [CrossRef]
23. Weng, P.; Zanuttini, B. Interactive Value Iteration for Markov Decision Processes with Unknown Rewards. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013.
24. Tetenov, A. Statistical treatment choice based on asymmetric minimax regret criteria. *J. Econom.* **2012**, *166*, 157–165. [CrossRef]
25. Ruszczyński, A. Risk-averse dynamic programming for Markov decision processes. *Math. Program.* **2010**, *125*, 235–261. [CrossRef]
26. Regan, K.; Boutilier, C. Robust Policy Computation in Reward-Uncertain MDPs Using Nondominated Policies. In Proceedings of the Twenty-fourth Aaai Conference on Artificial Intelligence, DBLP, Atlanta, GA, USA, 11–15 July 2011.
27. Kong, L.; Ye, L.; Wu, F.; Tao, M.; Chen, G.; Vasilakos, A.V. Autonomous Relay for Millimeter-Wave Wireless Communications. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 2127–2136. [CrossRef]
28. Sharma; Vishal; You, I.; Kumar, R. Energy Efficient Data Dissemination in Multi-UAV Coordinated Wireless Sensor Networks. *Mob. Inf. Syst.* **2016**, *2016*, 8475820. [CrossRef]
29. Gu, J.; Su, T.; Wang, Q.; Du, X.; Guizani, M. Multiple Moving Targets Surveillance Based on a Cooperative Network for Multi-UAV. *IEEE Commun. Mag.* **2018**, *56*, 82–89. [CrossRef]
30. Guo, S.; Jiang, Q.; Dong, Y.; Wang, Q. TaskAlloc: Online Tasks Allocation for Offloading in Energy Harvesting Mobile Edge Computing. In Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 16–18 December 2019.
31. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE ACM Trans. Netw.* **2016**, *2*, 2795–2808. [CrossRef]
32. Liu, T.; Zhu, Y.; Yang, Y.; Ye, F. Online task dispatching and pricing for quality-of-service-aware sensing data collection for mobile edge clouds. *CCF Trans. Netw.* **2019**, *2*, 28–42. [CrossRef]
33. Jeong, S.; Simeone, O.; Kang, J. Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning. *IEEE Trans. Veh. Technol.* **2018**, *3*, 2049–2063. [CrossRef]
34. Diao, X.; Zheng, J.; Wu, Y.; Cai, Y. Joint Trajectory Design, Task Data, and Computing Resource Allocations for NOMA-Based and UAV-Assisted Mobile Edge Computing. *IEEE Access* **2019**. [CrossRef]

35. Wang, L.; Huang, P.; Wang, K.; Zhang, G.; Zhang, L.; Aslam, N.; Yang, K. RL-Based User Association and Resource Allocation for Multi-UAV enabled MEC. In Proceedings of the 2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE, Tangier, Morocco, 24–28 June 2019.
36. Durga, S.; Mohan, S.; Peter, J.D.; Surya, S. Context-aware adaptive resource provisioning for mobile clients in intra-cloud environment. *Clust. Comput.* **2019**, *22*, 9915–9928.