applied sciences

MDPI

Article

Unmanned Aerial Vehicle Computation Task Scheduling Based on Parking Resources in Post-Disaster Rescue

Jinqi Zhu [1], Hui Zhao [2,*], Yanmin Wei [1], Chunmei Ma [1] and Qing Lv [1]

[1] School of Computer and Information Engineering, Tianjin Normal University, Tianjin 300387, China
[2] School of Cyberspace Security, Dongguan University of Technology, Dongguan 523106, China
* Correspondence: zhaoh@dgut.edu.cn; Tel.: +86-13929225512

Abstract: Natural disasters bring huge loss of life and property to human beings. Unmanned aerial vehicles (UAVs) own the advantages of high mobility, high flexibility, and rapid deployment, and are important equipment during post-disaster rescue. However, UAVs usually have restricted battery and computing power. They are not fit for performing compute-intensive tasks during rescue. Since there are widespread parking resources in a city, multiple parked vehicles working together to compute the applications from UAVs in a post-disaster rescue is investigated to ensure the quality of experience (QoE) of the UAVs. To execute uploaded task effectively, surviving parked vehicles within the monitoring range of an UAV are arranged into a cluster as much as possible. Then, the task execution cost is analyzed. Furthermore, a deep reinforcement learning (DRL)-based offloading policy is constructed, which interacts with the environment in an intelligent way to achieve optimization goals. The simulation experiments show that the proposed offloading scheme has a higher task completion rate and a lower task execution cost than other baselines schemes.

Keywords: disaster relief; deep reinforcement learning; unmanned aerial vehicles; task offloading; parking resources

Citation: Zhu, J.; Zhao, H.; Wei, Y.; Ma, C.; Lv, Q. Unmanned Aerial Vehicle Computation Task Scheduling Based on Parking Resources in Post-Disaster Rescue. *Appl. Sci.* **2023**, *13*, 289. https://doi.org/10.3390/app13010289

Academic Editor: Dimitris Mourtzis

Received: 18 November 2022
Revised: 8 December 2022
Accepted: 21 December 2022
Published: 26 December 2022

1. Introduction

Natural disasters bring great harm to people and damage the places where people live. As described in the literature [1], in recent decades, the number of natural disasters has increased significantly compared to before. In order to diminish the losses due to natural disasters, in addition to disaster prevention measures, valid post-disaster rescue is quite significant. Since science and technology have developed rapidly in recent years, unmanned aerial vehicles (UAVs) have become important equipment for post-disaster rescue [2]. Compared to traditional rescue instruments, UAVs own the advantages of fast mobility, small cost, easy installation, and flexible deployment, and they are able to simply access some disaster places that are previously inaccessible to humans. In addition, an UAV can be equipped with a variety of sensors, communication modules, and computing modules. The sensors can sense the disaster area and the sensed data can then be analyzed or delivered through wireless links. The combination of UAVs and the rapidly developing artificial intelligence (AI) science presents a new and efficient rescue mode for disaster rescue [3]. For example, in July 2021, due to heavy rainfall, many cities in Henan, China, suffered from serious floods. In this case, UAVs, which overcame adverse weather condition and cooperated with ground staff to carry out environmental reconnaissance and rescue missions, had achieved good rescue results.

Nevertheless, the participation of UAVs in disaster relief brings many challenges. Firstly, the battery power of an UAV is restricted because of the strict limitations on both the size and the weight of the UAV [4], which limits both the coverage range and the monitoring duration. Secondly, the processing power of an UAV is also not enough, which means that the UAV cannot complete AI applications, such as image processing, speech recognition,

and behavior recognition, in a timely manner. The reason is that AI applications, which generally demand complicated operations, are compute-intensive tasks. Besides, most of the AI applications often have response-time constraints. The resulting response timeout will render the task invalid. Hence, the application data from UAVs during a relief process should be uploaded in a timely manner, with the aim of prolonging the service duration of the UAVs.

Cloud computing, which is based on on-demand access via the Internet, is capable of offering reliable services for UAVs according to usage. This is because cloud computing offers various kinds of resources on demand. However, the geographical distance from the users to the cloud servers in the data center can be very long, which makes the quality of experience (QoE) of the users hard to be ensured. To assure a better QoE for UAVs, some studies described the transfer of data from UAVs to the edge server through multi-access edge computing (MEC) [5,6]. Compared to cloud-assisted methods, offloading using MEC servers is more appropriate for delay-sensitive and compute-intensive tasks. The computation capacity of a MEC server, however, is much more restricted than that of centralized clouds, and many tasks may face the problem of not being carried out in a timely manner once the workload of the MEC server is heavy [7]. To resolve the vital problem of designing an efficient uploading strategy for UAVs, reference [8] employed a large number of mobile unmanned ground vehicles (UGVs) to take part in UAVs' task calculation in a post-disaster rescue. However, unmanned ground vehicles are expensive [9], and the cost of employing a large number of UGVs to perform a task is very high.

Parked vehicles can be found everywhere in a city, and most automobiles have on-board units (OBUs) that provide computational and storage capabilities. Even if a disaster occurs, there are still some parked vehicles that are not damaged after the disaster. These parking resources can easily be utilized in task execution to satisfy the resource needs of UAVs. Therefore, we place emphasis on ground parked vehicles working together in order to efficiently perform the AI applications from UAVs. Specifically, to achieve a rational allocation of resources, undamaged vehicles that park outside within the monitoring range of a UAV are firstly formed into a cluster as much as possible. Then, we analyze the task execution cost, which considers both the time duration for finishing the task and the task performers' energy cost for performing the task. Finally, a deep reinforcement learning (DRL)-based uploading policy is constructed. By interacting with the environment in a smart manner, the task completion rate is maximized while the task execution cost is minimized. The simulation experiments show that our algorithm achieves a great improvement in task completion rate and task execution cost. The contributions of this paper are presented as follows:

Undamaged parked vehicles are arranged and then applied to help UAVs during a post-disaster rescue. The use of parked cars makes full use of a city's useful resources.

We arrange undamaged vehicles that park outside within the communication range of a UAV into a cluster as much as possible. The cluster architecture facilitates the management of parked vehicles and the allocation of the offloaded tasks.

Taking the advantages of DRL, we propose to build a task-scheduling model based on DRL to tackle the offloading issue of multiple tasks to meet the strict quality of experience (QoE) of UAVs.

The rest of this paper is organized as follows: Related works are discussed in Section 2. The motivating scenario is introduced in Section 3. Section 4 presents the task execution cost analysis, and we discuss the offloading solution in Section 5. In Section 6, we evaluate the performance of our scheme. Finally, the conclusions are presented in Section 7.

2. Related Work

Research on task offloading of UAVs can be divided into two types. Studies of the first type equip UAVs with process units to handle the tasks from user equipment. In [10], a UAV-enabled computing system is considered in which the UAV not only provides energy supply to end users, but also provides edge computing services for the uploaded data

from end users. The problem of how to maximize the MEC computing speed is studied. Yang et al. [11] investigate a network system with several UAVs, in which not only the UAVs but also the user equipment can perform tasks generated by the end users. With the aim of decreasing the energy cost of the UAVs, a method that jointly optimizes energy control, computing power allocation, and task execution delay is proposed. To provide sufficient computing resources for ground mobile devices, Zhang et al. [12] suggest that UAVs should equipped with powerful computing units. In [13], the authors believe that UAVs can be used as mobile clouds to complete tasks offloaded by mobile terminals with limited processing capabilities. The evaluation comparisons indicate the high achievement of the scheme presented in this paper. Liu et al. [14] propose that the tasks generated by sensor devices can be completed by the cooperation of their nearby idle sensors and the UAV installed with a MEC server. Hu et al. [15] introduce a system consisting of an access point, an UAV, and many ground user devices. UAVs can perform the tasks uploaded by user equipment and can also relay task data to access points for processing. Inspired by the fact that UAVs can communicate with each other, Ref. [16] proposes a UAV-based wireless energy transfer system. However, because the energy power of the UAV is restricted, a continuous growth in the number of tasks leads to a continuous decline in the success ratio of task execution.

Research studies of the second type aim to execute tasks with the help of remote clouds or end users' nearby edge servers. For example, in [17], data are transmitted between the terminal and the access device via an UAV. The UAV can also transmit task data from the source device to the edge for timely processing. Ref. [18] combines edge computing with UAVs. Security problems in the task execution process are presented. The authors in [7] describe an energy-saving computation offloading scheme for UAV-MEC systems with emphasis on physical-layer security. The authors in [19] combine an UAV with edge computing. The UAV collects data from Internet of Thing (IoT) devices and later transmits data to access points, which perform their best to compute and process data. To minimize both the energy exhaustion of IoT devices and the energy spent on data offloading of the UAV, Ref. [20] proposes a UAV-aid MEC paradigm. The experimental results indicate the paradigm remarkably reduces the energy utilization compared to other existing strategies. To optimize the task execution latency of the fog computing system with one UAV, Li et al. [21] present a scheduling algorithm and a multi-task offloading scheme under a multi-server environment. The authors in [22] jointly optimize computational task offloading and resource assignment. The authors in [23] suggest that the data collected by UAVs can be transferred to the edge server which is connected to the base station via high-speed wired links. A smart uploading mechanism specifically developed for the UAV-assisted MEC network was designed. In [24], to obtain good system utility, the authors propose that a single application can be split into multiple tasks and uploaded to the server node for further analyzing.

To facilitate disaster rescue, the authors in [25] present a network consisting of two subnetworks, which are an aerial subnetwork formed by multiple UAVs and a ground vehicular network formed by automobiles. Three kinds of communications are introduced to deliver data efficiently. Ref. [8] describes the use of mobile unmanned ground vehicles (UGVs) to perform the tasks of the UAVs in a post-disaster rescue application. Since unmanned ground vehicles are expensive, they cannot be applied to post-disaster rescue scenarios on a large scale. Moreover, there are many vehicular applications based on parked vehicles. In [26], owning to the high contact opportunities with parked cars, the authors let outside parking distribute multimedia data via Internet of Vehicles (IoV). In [27], the authors discuss how to make use of computational resources from parked cars to execute mobile applications for moving vehicles. In [28], the authors arrange parked vehicles to perform real-time AI tasks, with the aim of meeting the service quality of vehicle users. The authors in [29] propose that both parked vehicles and their neighboring mobile cars on the road are able to form vehicular social groups in the city. Efficient content transmission is achieved based on the formed groups. In this paper, undamaged vehicles that park

outside are rationally arranged to take part in data computation. Furthermore, based on the superiority of DRL, we construct a DRL model to tackle data offloading from UAVs, with the aim of minimizing the desired optimization goals.

3. Motivating Scenario

3.1. System Model

The system model in a post-disaster rescue is presented in Figure 1, which mainly includes two entities: UAVs and exteriorly parked vehicles that have not been damaged after a disaster.

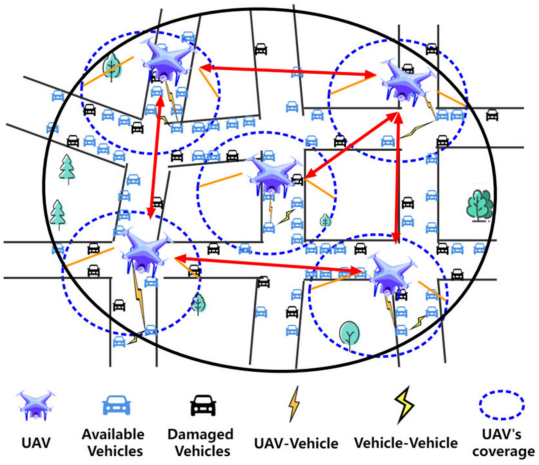


Figure 1. System model in post-disaster rescue.

UAV: Suppose the set of UAVs in the monitoring area is $U = \{1, 2, 3, \ldots, u\}$. After taking off from a random position on the ground, each UAV selects a position at a height H from the ground and hovers horizontally at this position for area monitoring. H denotes the minimum altitude fit for area monitoring to prevent recurrent going down and going up. Suppose the coverage radius of each UAV is R_0, and the coverage regions of the UAVs do not overlap with each other. Multiple UAVs can be organized into a UAV ad hoc network, and each UAV can transfer data to other UAVs through the network. Suppose that the communication radius of the UAV in the formed ad hoc network is R. According to [30], we have $R_0 < R$. The UAVs aim to monitor the disaster area and obtain information from the target area at the first time.

Parked vehicle: vehicles that park outside can be found everywhere in a city. Parked vehicles have the characteristics of short radio range, low bandwidth, limited storage and computational capacities, and wide distribution. Moreover, since parked vehicles are relatively stable, the end user's contact time with a parked vehicle is usually much longer than with a moving vehicle. Hence, compared to mobile vehicles, parked vehicles are more suitable for uploaded task execution. Suppose the set of parked vehicles in the monitoring area is $N = \{1, 2, 3, \ldots, n\}$. Although a number of vehicles parked on the ground have been damaged in the disaster, a large number of these vehicles have not been damaged. These undamaged ground parked vehicles can serve as natural roadside computing facilities, and supply data offloading and computing services for UAVs in a collaborative manner.

Cluster: There may be multiple parked vehicles in the monitoring area of the UAVs. To ease the management of the parked vehicles and optimize the allocation of parking resources, undamaged ground parked vehicles form clusters according to the method presented in Section 3.3. The clusters aim to offer computing service and even storage service for UAVs if necessary.

Moreover, we assume parked vehicles and UAVs are equipped with OBUs through which they can send data to one another. UAVs are also equipped with another communication model (such as IEEE 802.11*n*) to enable a high rate of task data delivery among the

UAVs. There are four types of wireless communication in our proposed system architecture: (1) UAV to parked vehicle; (2) parked vehicle to UAV; (3) UAV to UAV; and (4) parked vehicle to parked vehicle. The first two types of communication can occur if a parked vehicle is found to be within the communication range of an UAV. An UAV can deliver its packets to another UAV through the UAV ad hoc network. Furthermore, parked vehicles are able to transfer data to each other by using the IEEE 802.11*p* protocol.

3.2. Disaster Model

In this subsection, which sums up the familiar characteristics in many disasters, a simplified disaster model is shown as follows: suppose a disaster event i is a tuple of four parameters (C_i, K_i, I_i, and A_i), where C_i denotes the central position of the area that the disaster event happens, as described by two-dimensional coordinates (x_i, y_i); K_i is disaster intensity, which is used to express the risk level of the disaster event at C_i; I_i represents the attenuation coefficient of this event; and Ai is the affected area of this disaster event. As described in [31], a disaster is described by a group of disaster events (devastating events). For each devastating event i, I_i denotes the change of the risk level (intensity) in the affected area of event i. Apparently, the closer a location is to the center point of the devastating event, the greater its disaster intensity. We assume I_i decreases linearly from the central position of the event. For location j, the attenuation coefficient is calculated as follows:

$$I_{i,j} = \frac{l_j}{L} \tag{1}$$

where L is the farthest distance from C_i to the location affected by the devastating event in the monitoring area, and l_j is the distance from C_i to location j. For parked vehicle n located at location j, its survival probability can be expressed as follows:

$$p(n) = \frac{I_{i,j} \times K_i}{K_{\max}} \tag{2}$$

where $K_{\max}$ denotes the highest recorded intensity of such devastating events. For example, for earthquake disasters, the highest earthquake magnitude achieves 12. As shown in Figure 2, based on the survival probabilities of the parked vehicles, we categorize ground parked vehicles into different states: safe, sub-safe, and dangerous. We assume there are two thresholds, T_s and T_d (T_s < = T_d), which can be predefined in terms of the disaster scene. Let $rank(n)$ denote the state of parked vehicle n, we then obtain the following:

$$rank(n) = \begin{cases} safe\ p(n) < T_s \\ sub\ safe\ T_s \le p(n) \le T_d \\ dangerous\ T_d \le p(n) \end{cases} \tag{3}$$

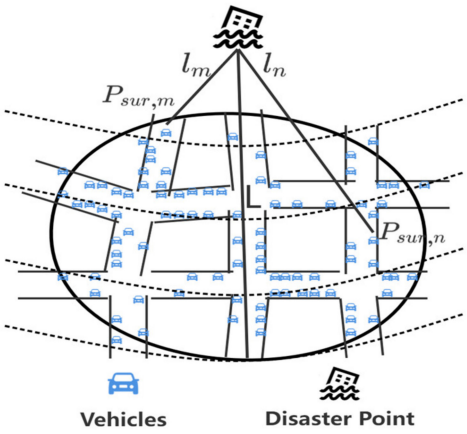


Figure 2. Disaster model.

Compared to vehicles in a dangerous state, ground vehicles in a safe state and a sub-safe state are more suitable for offloaded task execution because they are relatively far away from the center of the disaster event and have less damage. Suppose the cluster head periodically sends beacon messages to the UAV, from which the UAV learns the current state of each member node, its ID number, and the total number of surviving parked cars in the cluster.

3.3. Cluster Construction

To execute the offloaded task efficiently, the surviving parked vehicles within the monitoring range of a UAV are formed into a cluster as much as possible. The establishment process of the cluster includes the following steps:

1. Each UAV sends its ID number to the parked vehicles within its coverage range. The surviving parked vehicles then record the received UAV ID information.

2. A value between 0–1 is randomly chosen by each parked vehicle. For parked vehicle n, if the selected value r_n satisfies the condition that:

$$r_n \leq \frac{p}{1 - p(k \bmod \frac{1}{p})} \tag{4}$$

then parked vehicle n becomes the cluster head. p in Equation (4) is the percentage of the number of cluster heads to the total number of parked vehicles, and k denotes the current number of rounds. Equation (4) aims to enable each node in the cluster to take turn to become the cluster head, with the aim of balancing the load.

3. After the cluster head is determined, the cluster head transmits a broadcast message including its own ID number and its recorded UAV ID to the remaining non-cluster head nodes. To reduce the delivery overhead of the broadcast message, the dissemination hops of the broadcast message is specified as M_h. The remaining non-head vehicles decide whether to join the cluster according to the ID number of the UAV the cluster head belongs. Specifically, for a parked vehicle, if it receives multiple broadcast messages from different cluster heads, and the UAV IDs of these broadcast messages are the same, the vehicle randomly selects a cluster to join in. If the UAV IDs of the received broadcast messages are different, the parked vehicle joins the cluster whose cluster head owns the same UAV ID as this parked vehicle. Our aim is to try to keep parked vehicles within the same UAV coverage range in the same cluster.

4. Repeat steps 2 and 3 if there are still parked vehicles that have not been added to any cluster. After the above steps, there may still be some isolated vehicles that have not joined any cluster.

5. When the cluster is too small, the total resources of the cluster may be insufficient for data processing, so that it is necessary to merge several small clusters into a larger cluster. For the cluster whose number of members is no more than the threshold C_{th}, some cluster heads of these clusters are randomly selected as the upper-level cluster heads according to step 2. After the upper-level cluster heads are determined, small clusters are merged according to step 3.

6. Continue to perform step 5 until the remaining small clusters cannot be merged. Figure 3 shows the cluster establishment process.

After the cluster is formed, each vehicle member in the cluster periodically sends a message, including its vehicle ID number, its current location coordinates, its remaining energy power, and its current working status, to the cluster head. If no message is received from a member node for some time, the head node considers the node to be corrupted or no long in the cluster.

The advantages of the cluster are as follows: (1) Message binding: small messages delivered to the same terminal point can be bundled into a larger message by the head node to decrease the data delivery overhead. (2) Vehicle management: parked vehicles can be well managed in the cluster. Moreover, computing resources in the cluster can be allocated uniformly and fairly within the cluster. (3) Results merging: multiple final results

with small amounts of data can first be merged in the cluster and then returned back to the UAV to reduce the consumption of network resources. In addition, since the probability of a parked vehicle leaving or joining the cluster in a disaster scenario is considerably lower than in a normal urban situation, we believe the structure of the cluster is relatively stable.

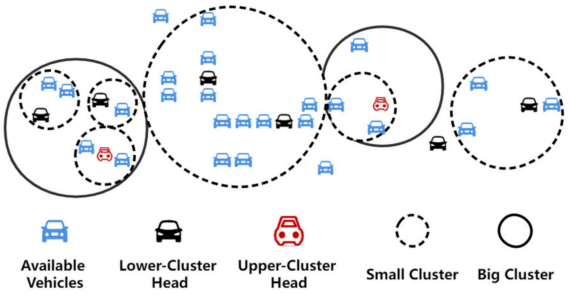


Figure 3. The formation of the clusters.

4. Task Execution Cost Analysis

4.1. Task Execution Latency

In this subsection, the task execution time, which is a part of task execution cost, is analyzed. For UAV u, when there are sufficient parking resources in its coverage area, the tasks from UAV u will be directly delivered to the parked vehicles in its coverage range. The time cost for finishing task t from UAV u is calculated as follows:

$$Tt = T_{u,t}^{up} + T_{u,t}^{com} + T_{u,t}^{down} \tag{5}$$

where $T_{u,t}^{up}$ denotes the time to upload the whole task; $T_{u,t}^{com}$ indicates the task calculation time; and $T_{u,t}^{down}$ indicates the result return time. For $T_{u,t}^{up}$, we have the following:

$$T_{u,t}^{up} = \frac{s_{u,t}}{mr_{u,ave}} \tag{6}$$

where $s_{u,t}$ is the total amount of data size; $r_{u,ave}$ is the average throughput between UAV u and the parked vehicles which will carry out the task; and m denotes the number of task performers. Here, we consider a parallel task execution mode [32,33]. In other words, the whole task can be divided up and then carried out by multiple task performers collaboratively. The throughput between UAV u and parked car n is modeled as follows:

$$r_{u,n} = B_{u,n} \log_2(1 + \frac{P_u |h_{u,n}|^2 d_{u,n}^{-\beta}}{\sigma^2}) \tag{7}$$

where $B_{u,n}$ is the channel bandwidth; P_u denotes the fix transmit power of UAV u; σ^2 is the power spectral density of Gaussian Noise; $h_{u,n}$ indicates the channel fading from UAV n to parked vehicle n; $d_{u,n}$ represents the distance between the two when a downlink delivery occurs; and β indicates the path loss exponent. Assuming that the CPU frequency of the automobile is f_v, then $T_{u,t}^{com}$ is denoted as follows:

$$T_{u,t}^{com} = \frac{s_{u,t} c_{u,t}}{m f_v} \tag{8}$$

where $c_{u,t}$ indicates the computational complexity of task t. The task result return duration is calculated as follows:

$$T_{u,t}^{down} = \frac{y_{u,t}}{r_{ave,u}} \tag{9}$$

where $y_{u,t}$ is the size of the final result, and $r_{u,ave}$ which denotes the average transmission rate from task performers to UAV n has a similar calculation method as $r_{u,n}$.

If the parking resources within the coverage range of UAV u are insufficient (computing resources are occupied) or there is no available resource in the coverage range of UAV u, UAV u needs to forward the generated application to its neighboring UAV with more accessible resources. In this case, the required duration for task calculation is as follows:

$$T_t = T_{u,u'} + T_{u',t}^{up} + T_{u',t}^{com} + T_{u',t}^{down} + T_{u',u} \tag{10}$$

where $T_{u,u'}$ is the duration for UAV u to forward the data to its neighbor UAV u' through the wireless mode, and $T_{u',u}$ denotes the wireless delivery delay for UAV u' to upload the calculation result to UAV u. Since two neighboring UAVs communicate through an air-to-air (A2A) link which mainly uses line-of-sight transmission [26], $T_{u,u'}$ is described by the following equation:

$$T_{u,u'} = \frac{s_{u,t}}{r_{u,u'}} = \frac{s_{u,t}}{B_{u,u'} \log_2(1 + \frac{P_u 10^{\frac{-Loss(u,u')}{10}}}{\sigma^2})} \tag{11}$$

where *Loss(u,u')* is the path loss between the two UAVs in free space. According to study [34], we have the following:

$$Loss(u, u') = 32.45 + 20 \log f_f + 20 \log d_{(u,u')} \tag{12}$$

where f_f represents the carrier frequency, and $d(u,u')$ is the distance between the two neighboring UAVs. Moreover, $T_{u',u}$ is obtained in a similar way as $T_{u,u'}$. $T_{u',t}^{up}$, $T_{u',t}^{com}$ and $T_{u,t}^{down}$ denote the duration for the task performer to download the task, the time used to perform the task, and the time to send back the result, respectively, which are calculated using the same method as $T_{u,t}^{up}$, $T_{u,t}^{com}$, and $T_{u,t}^{down}$ respectively.

4.2. Energy Comsumption

Since the battery strength of an automobile is restricted, the energy overhead for data uploading of a parked car must be taken into account. Since the energy consumption of a vehicle occurs during the task receiving, sub-task performing, and task result transmitting processes, the total energy overhead for executing task t is expressed as follows:

$$E_t = \sum_{t=1}^{m} P_{v,\text{send}} T_{u,t}^{up} + \delta f_v^2 T_{u,t}^{com} + P_{v,rec} T_{u,t}^{down} \tag{13}$$

where $P_{v,\text{send}}$ and $P_{v,rec}$ denote the delivery power and the receiving power of the vehicle, respectively; δ is a coefficient that relies on the chip architecture of the CPU; and m is the set of task performers. Considering both the time cost for finishing a task and the energy cost of task execution by parked vehicles, the total task execution cost of task t is modeled as follows:

$$\cos t_t = \lambda T_t + (1 - \lambda) E_t \tag{14}$$

where λ represents the weight of the time spent on completing the task.

5. Data Offloading Solution

In this section, the optimal resource assignment is firstly discussed, and then deep reinforcement learning (DRL) is adopted to explore an optimal offloading policy.

5.1. Estimating the Amount of Resources

As the amount of resources allocated to each task directly determines the efficiency of task completion, resource allocation becomes a critical issue in high-performance uploading mechanism devising. Our goal is to minimize the task execution cost. Since the duration for task performers to perform a task is critical to both the time used for finishing the task and the energy expenditure of the vehicles, the total task completion duration should be firstly

optimized when assigning the CPU resources of the vehicles. Therefore, the minimization problem is formulated as follows:

$$\begin{aligned} P: \min \sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{m=1}^{f_{\max}} x_{u,t}^{m} \frac{s_{u,t} c_{u,t}}{f_{u,t}^{m}} \\ s.t.\ \mathrm{C1}: x_{u,t}^{m} \in \{0,1\}, \forall u \in U, t \in T, m \leq f_{\max} \\ \mathrm{C2}: \sum_{m=1}^{f_{\max}} x_{u,t}^{m} = 1, \forall u \in U, t \in T, m \leq f_{\max} \\ \mathrm{C3}: \sum_{u=1}^{U} \sum_{t=1}^{T} \sum_{m=1}^{f_{\max}} f_{u,t}^{m} \leq R_{all} \end{aligned} \tag{15}$$

where $f_{\max}$ is a pre-defined maximum allocation resource threshold for each task, and $x_{u,tm}$ is resource choice profile. In P, constraints C1 and C2 guarantee that the resource selection of each task is binary. C3 limits the tasks to be assigned from the UAVs within the resource permissible range. Let the resource set of each task be $R_{set} = \{0.1, 0.2, \ldots, f_t, f_{max}\}$ and let the maximum weight of the owned parking resources within the monitoring area of the UAVs be R_{all}, P is then changed into obtaining the minimum value of the multi-dimensional knapsack issue, which is tackled here by dynamic programming [35].

5.2. Task Offloading Decision

Once given a certain resource value, the number of task splits is obtained. For example, if the amount of computing resources assigned to task t is f, this task has $\lceil f/f_v \rceil$ sub-tasks that are executed simultaneously. Currently, DRL, which is an extension of the customary reinforcement learning (RL), has been universally applied to many optimization issues. By exploring and receiving feedback from the environment, and combining with extensive state-action space exploration, DRL is good at finding the optimal offloading strategy. Using DRL for task allocation provides a main advantage. That is, during the learning phase, DRL learns the reward function which can be reasonably defined to capture multiple optimization objectives. Because of the advantages of DRL, DRL is adopted to identify specific task performers in this subsection. The key definitions of DRL are modeled as follows:

Agent: An agent is able to select the next action based on the current state of the environment and make decisions through continuous interaction with the environment according to the observations. In the proposed UAV task offloading scenario, the cluster head in each UAV's monitoring scope plays the role of the agent. The agent is responsible for gathering information and studying the task placement strategy iteratively, so as to maximize the cumulative reward.

State: Discovering a suitable representation of the input state is important to DRL model. Since the task uploading decision in our paper relies on the maximum tolerance delay, the data size of each task, and the locations of the parked cars in the cluster, the system state of the proposed DRL is represented as a vector $S_t = (s_t, l_1, l_2, \cdots, l_n, T_t^{\max})$, in which s_t is task data size, $T_t^{\max}$ indicates the maximum tolerance delay threshold, and l_1, $l_2, \ldots, l_n$ is the locations of the member nodes in the cluster within the coverage range of the UAV.

Action: The cluster head makes the decision on which ground vehicles are allocated to the uploaded tasks. We use vector $A_t = (a_{t1}, a_{t2}, \cdots, a_{tM})$ to represent the actions taken in state S_t, in which $a_{ij} \in \{0,1\}$ for $\forall j \in \{1,2,\cdots,M\}$. $a_{ij} = 1$ denotes that vehicle n is chosen as the task performer. Different from traditional DRL, we have $\sum_{j=1}^{M} a_{ij} > 1$. The reason is that, for each task, multiple candidates are chosen for task computing. This is because one vehicle does not have enough resources to meet the task delay requirement.

Reward: At each state, the agent takes an action to receive the immediate reward. Our goal is to obtain the minimal task execution cost, while the goal of the DRL model is to

maximize the reward. In order to retain consistency with the proposed minimization target, the reward function is defined as follows:

$$r_t = \begin{cases} \frac{\tau * rate}{\lambda T_t + (1-\lambda) E_t} & T_t \leq T_t^{\max} \\ 0 & T_t > T_t^{\max} \end{cases} \tag{16}$$

where *rate* denotes the current task's successful ratio, and τ is a coefficient which is exploited to avoid slow gradient descent speed due to a small reward value.

Policy: Compared to traditional RL, neural network which brings various observations into the offloading policy is introduced into DRL. The output of the *softmax* function of the neural network is taken as the policy in our learning-based model. The policy is presented as follows:

$$\pi_\theta(s_t, a_{tn}) = P(a_{tn}|s_t, \theta) = \frac{e^{\phi(s_t, a_{tn})\theta}}{\sum_{j=1}^{M} e^{\phi(s_t, a_{tn})\theta}} \tag{17}$$

This policy is used to determine the probability of using parked vehicle n to complete task t in state s_t.

The objective function: the optimization is based on the following equation:

$$J(\theta) = -E_{\pi\theta}[Q_\pi(s_t, a_t) \log \pi_\theta(s_t, a_t)] \tag{18}$$

where $Q_\pi(s_t, a_t)$ indicates the value acquired when taking action a_t under state s_t. We obtain the following:

$$Q_\pi(s_t, a_t) = \begin{cases} r_t + \gamma Q_\pi(s_{t+1}, a_{t+1}) & s_t is\ termal\ state \\ r_t & otherwise \end{cases} \tag{19}$$

where γ is the attenuation factor, and r_t is the reward received by taking the action at state S_t. Algorithm 1 summarizes the task performer selection process based on DRL.

Algorithm 1: Task Performer Allocation Based on DRL.

1. Input:*iteration number*, *step*, *batch-size*, the action set A, attenuation factor γ
2. Output: $A_t = (a_{t1}, a_{t2}, \cdots, a_{tM})$
3. for episode from 1 to *iteration number*
4. Initialize s_0 and then obtain the feature vector $\phi(s_0)$;
5. for t from 1 to *step*
6. chose action a_t according to constrain (17);
7. take action a_t, obtain reward value and alter st by $st \leftarrow s_{t+1}$;
8. store $\{\phi(s_t); a_t; s_t; \phi(s_{t+1});\}$ in memory;
9. if s_{t+1} is the terminal state
10. end the iteration;
11. else
12. continue;
13. end for
14. sample *batch-size* of transitions randomly for training;
15. update the *objective function* according to the gradient descent;
16. end for

6. Simulation Result

In this section, evaluations are conducted to present the performance of the scheme proposed in this paper. *Pytorch* is applied to build the deep learning framework. In the simulation scenario, we consider a circular area consisting of four UAVs and a number of parked vehicles, which number varies from 100 to 500 [36]. Suppose the coverage radius of each UAV is 200 m, and the UAVs are located at a fixed height of 10 m [37]. Each UAV randomly generates tasks with data size from 2 Mbit to 4 Mbit. The maximum delay tolerance of the tasks ranges from 4 s to 8 s. The size of the final result is considered as

0.1 Mbit. The transmission range of the vehicle is denoted as 300 m. The local default CPU frequency of the vehicle is set as 0.4 GHz. The communication radius between two neighboring UAVs is 300 m. Other parameter values in evaluation are described in Table 1. Moreover, two hidden layers are constructed in the neural network used in the proposed DRL model. The numbers of neurons in these two hidden layers are 512 and 256, respectively. The activation function used is Rectified Linear Activation Function(Relu).

Table 1. The setting of variable values.

Parameter	Value	Description
$B_{u,n}$	10 MHz	the bandwidth from UAV u to vehicle n
P_u	10 W	the transfer power of UAV u
P_n	1 W	the fixed transfer power of vehicle n
$B_{u,u'}$	50 MHz	the bandwidth between neighboring UAVs
σ^2	−95 dBm	the power spectral density of Gaussian Noise
β	2	path loss exponent
f_f	2 GHZ	carrier frequency
τ	100	coefficient according to gradient descent
λ	0.6	weight value
batch-size	80	the number of units manufactured in a production run
γ	0.8	attenuation factor
step	80	step in the DRL model
learning rate	0.0001	learning rate in training

For comparisons, other two task offloading schemes are presented to validate the performance of the scheme described in our paper, where local computing (LC) places all the tasks in the UAVs for computing and in random offloading scheme (RF), each UAV randomly selects parked vehicles for task uploading, while the computational resources are allocated based on the strategy presented in Section 5.1.

6.1. Evaluation Results of the Learning Rate and the Attenuation Factor

The learning rate is critical to maximize the speed of learning. A proper learning rate could let the objective function converge to the optimal value at the right time. Hence, the effects of the proposed model at different learning rates are studied and shown in Figure 4a. In Figure 4a, when the learning rate is 1×10^{-6} and 1×10^{-5}, the task completion rate slowly converges with the increase in iteration numbers. However, the model cannot achieve the optimal value, but the local optimal value, and no real optimal solution is found. When the learning rate is 0.001, the model converges very fast in the earlier stage, but the successful rate fluctuates too much to converge with the iteration process. When the learning rate is 0.0001, both the optimization rate and the stability of the model are guaranteed. Therefore, the value of learning rate is chosen as 0.0001 in the process of model tuning.

Attenuation factor is a key element of the DRL model. The value of the attenuation factor is usually between 0 and 1. This is because the current reward and all subsequent rewards have to be considered in DRL. The training effect of the proposed DRL model under different attenuation factors is shown in Figure 4b, from which we know that when the attenuation factor is 0.8, the model achieves the best performance compared to other attenuation factor values.

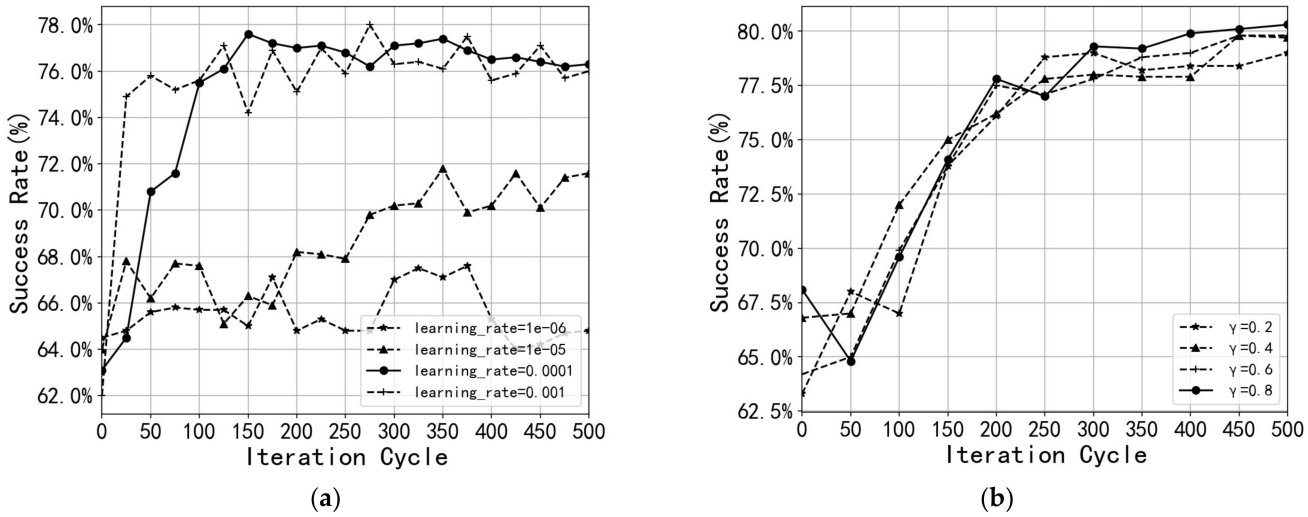


Figure 4. Evaluation under different parameters of training. (**a**) Task successful rate under various learning rates, and (**b**) task successful rate under various attenuation factors.

6.2. Evaluation Results of Modifying Computational Complexity

In this subsection, the two metrics in term of computational complexity are discussed, in which the computational complexity denotes the CPU frequency demand for computing a unit bit of data. Figure 5a,b present the successful ratio and the task execution cost of the schemes involving uploading. Figure 5a shows that, when increasing the computational complexity, the task completion rates decrease for all schemes. This is in conformity with our intuition. We can observe that our learning-based scheme performs the best, while the LC scheme is the worst in performance. When the computational complexity is set to 1200 rounds/bit, the task completion rate is only 8% for the LC scheme and 32% for the RF scheme, while the successful rate for the proposed DRL based scheme in this paper is 64%. Compared to the RF scheme and the LC scheme, the task successful rate of our scheme increases by 56% and 32%, respectively. The reasons are as follows: Firstly, computational resources are reasonably allocated by the resource assignment strategy discussed in this paper. Secondly, both the features of flexibility and long-term planning of DRL are fully utilized in our scheme. Moreover, when the computational complexity is 900 rounds/bit, the successful rate of the proposed scheme is up to 83%, which is much higher than other two schemes involving uploading.

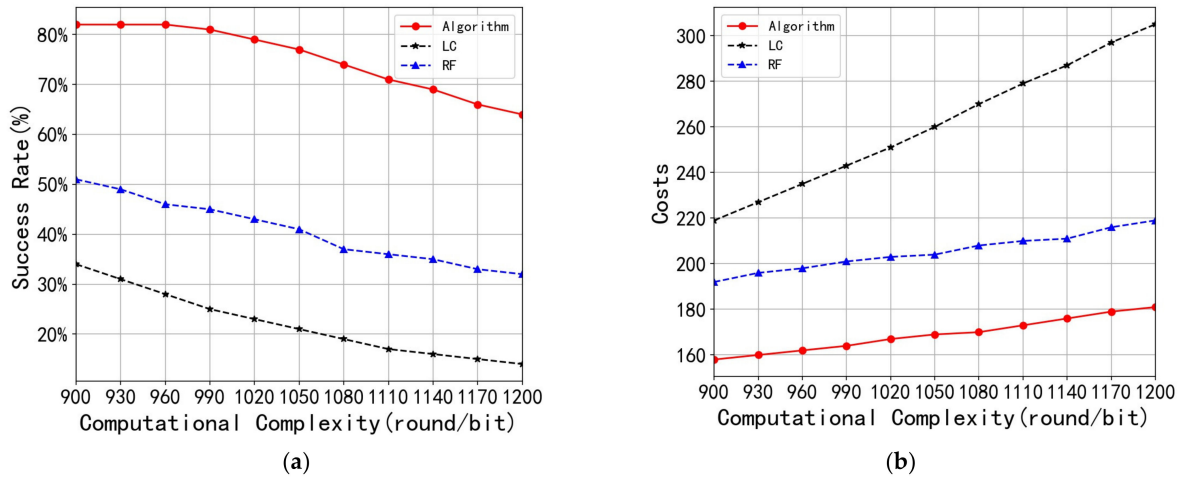


Figure 5. Simulation under diverse computational complexity values. (**a**) Task successful rate under various computational complexity, and (**b**) execution cost under various attenuation factors.

From Figure 5b, we can see that the task execution cost of the LC scheme tends to increase linearly when the values of the computational complexity are modified due to both the increase in task execution time and the increase in energy consumption. The proposed scheme obtains great advantage on the total task execution cost even when the computation complexity is set to a high value. The main reason is that, in the proposed scheme, task offloading can be continuously optimized through multiple iterations to maximize the reward by using the DRL model.

6.3. Evaluation Results of Modifying the Generated Tasks

This subsection discusses the effect of the number of tasks to be processed on the performance of these schemes. When the computational complexity is set as 1000 rounds/bit, the evaluation results of each offloading scheme are presented in Figure 6a,b as the number of tasks is gradually increased from 40 to 80. The successful rates for these three offloading methods decline as we increase the number of tasks. It could be expounded by the fact that, once the number of compute-intensive tasks from the UAVs is small, parking resources are sufficient for the assignment. However, since the more tasks there are, the less CPU resources per task can be obtained with a fixed amount of deployed resources. The successful rates of the three schemes reduce significantly with a large number of tasks. Because of the advantages of the DRL model, the proposed learning-based scheme shows a higher successful rate than other two schemes.

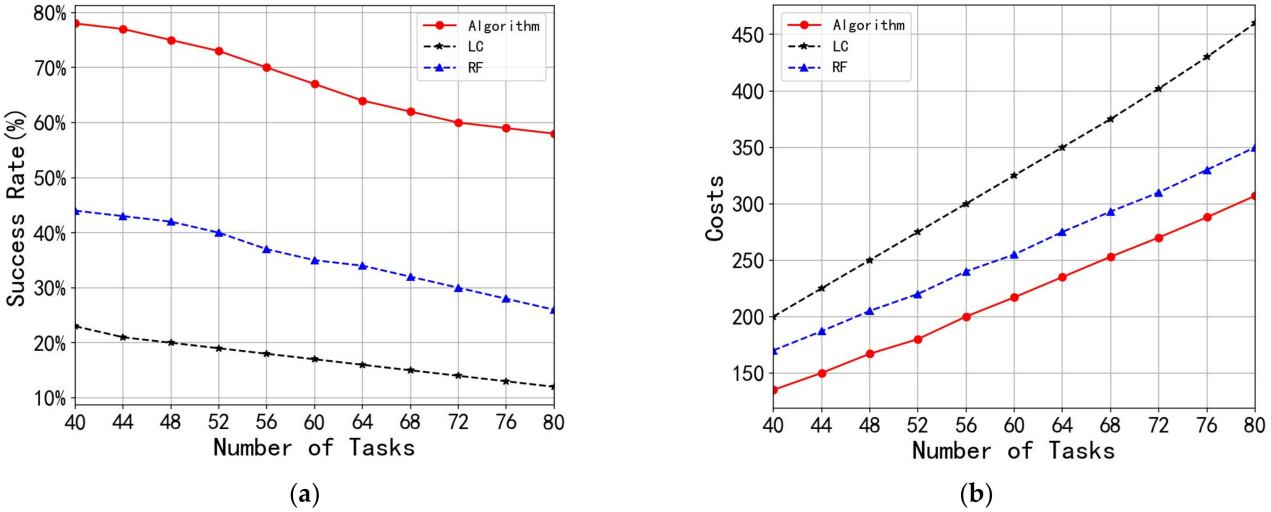


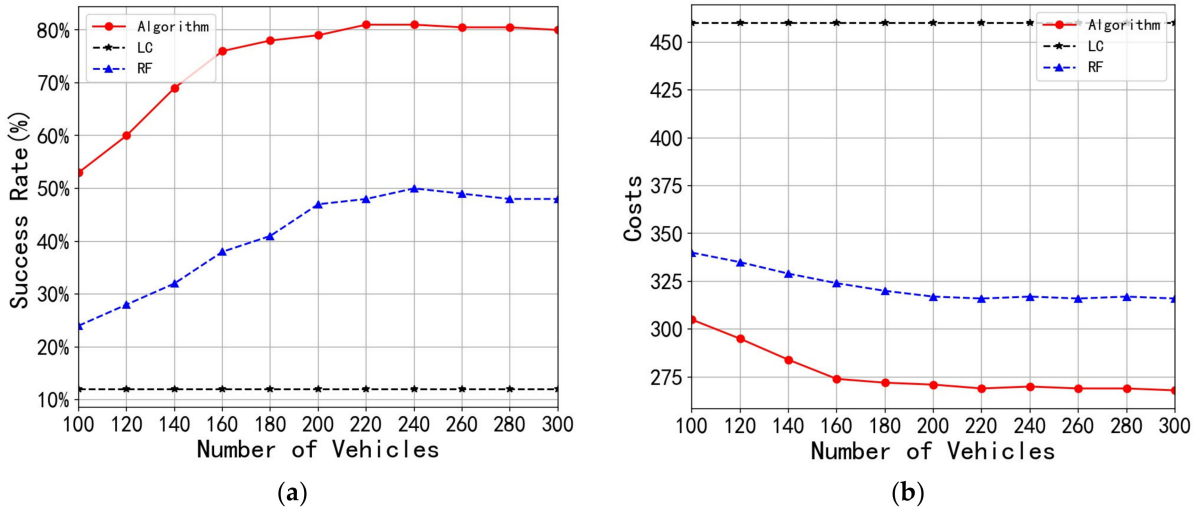
Figure 6. Simulation results with diverse number of tasks. (**a**) Task successful rate under diverse number of tasks, and (**b**) execution cost under diverse number of tasks.

Figure 6b shows our scheme greatly decreases the task execution cost. The reason is that the proposed DRL-based scheme continuously adjusts the offloading strategy through multiple training processes to achieve the lowest task execution cost, with a relatively high task successful rate, among the three schemes.

6.4. Evaluation Results of Modifying the Amount of Parking Resources

In Figure 7a, we focus on the task completion rate when modifying the parking resource value in the disaster area, in which the amount of resources depends on the number of available parked vehicles. The LC scheme and RF scheme are also set as the benchmarks. It is obvious that the proposed scheme is significantly superior to both of the comparison schemes. When increasing the amount of available resources, the gap is even more apparent. Since the tasks are only performed locally, the curve of task completion time remains stable for the LC scheme. Once the number of ground parked cars reaches 200, the successful rate of the proposed learning-based scheme is 79%, while the successful

rate of the other two schemes are 9% and 44%, respectively. We can conclude that, under the same conditions, the proposed learning-based scheme can accomplish more tasks.



(**a**) (**b**)

Figure 7. Simulation results with diverse deployed resources. (**a**) Task successful rate under resources, and (**b**) execution cost under resources.

Figure 7b certificates that the total task execution cost of our method is much smaller that of the traditional LC and RF schemes. The reason is that an increase in the number of parked cars results in an increase in the amount of available resources, which shortens the time consumption for executing a task. Thus, the total execution cost of the task is reduced.

6.5. Evaluation Results of Modifying λ and Evaluation Test on Task Allocation Time

Since the weight λ in Equation (14) affects the total task execution cost, we evaluate the execution cost of the three schemes under different value of λ. From Figure 8, we know that, when changing λ, our proposed scheme always achieves the lowest task execution cost among the three schemes. Furthermore, to verify the efficiency of task allocation, we test the task allocation time required to allocate 50, 100, 150, and 200 tasks, respectively, with the results being shown in Figure 9. As can be seen from this figure, the proposed DRL-based scheme takes a short time to complete task assignment, which indicates the efficiency of the proposed DRL model. For example, the total allocation time for assigning 100 offloaded tasks is only 0.057 s.

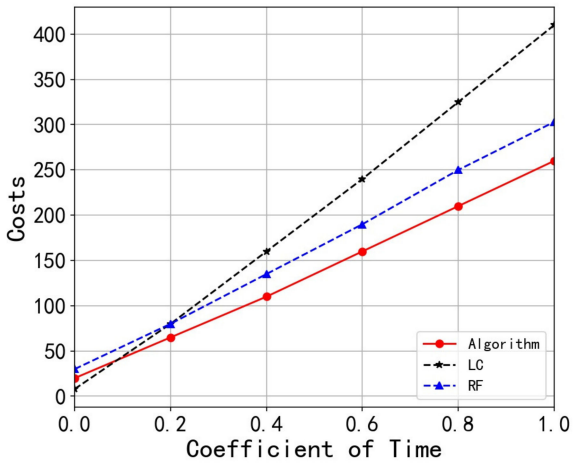


Figure 8. Simulation results with diverse λ.

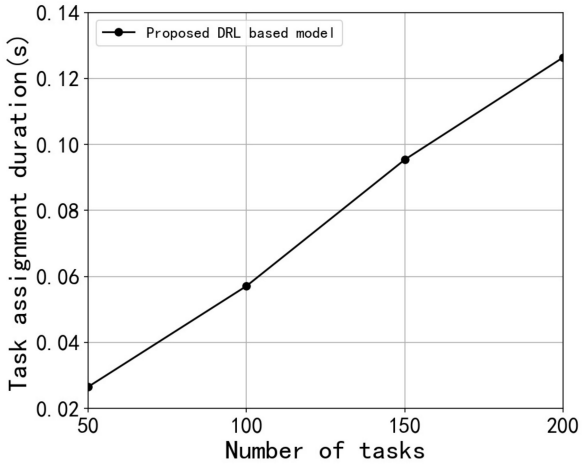


Figure 9. Task assignment duration using our proposed model.

6.6. Evaluation Results of Modifying C_{th}

In this subsection, we also test the task successful rate and task execution cost of our scheme with diverse C_{th} value, which is discussed in Section 3.3. In Figure 10, we see that, when C_{th} increases from 80 to 200 parked vehicles, the successful rate increases while the task execution cost decreases. The main reason is that an increase in the number of cluster members results in an increase in the amount of available computational resources in the cluster, which shortens the time consumption for executing the tasks.

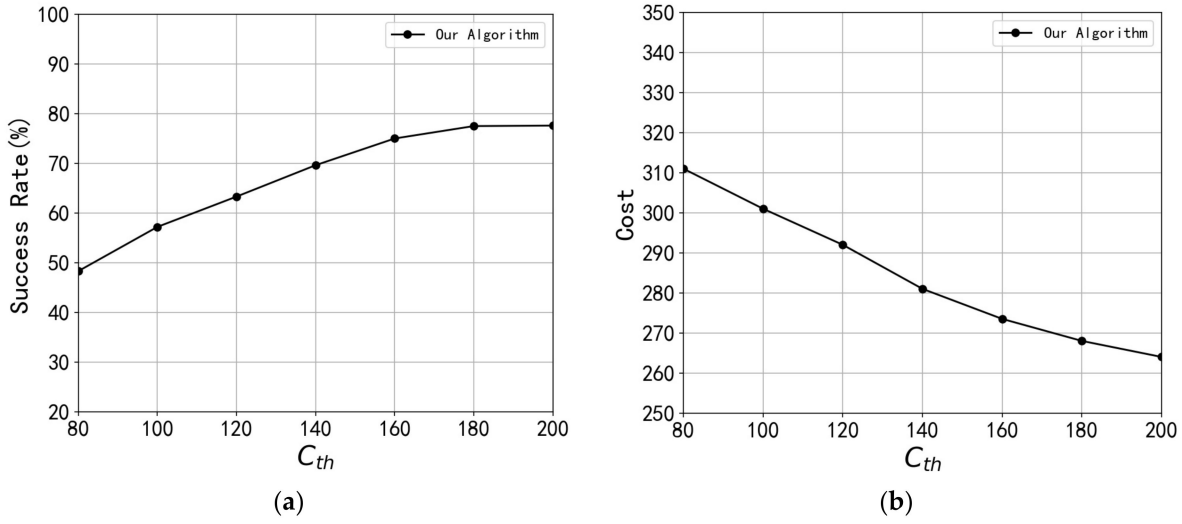


Figure 10. Simulation results with diverse value of C_{th}. (**a**) Task successful rate under C_{th}, and (**b**) execution cost under C_{th}.

7. Conclusions

In post-disaster scenarios, infrastructures are damaged, but some parked vehicles are not. Because parked vehicles have computational resources, this paper places emphasis on reasonably organizing undamaged ground parked vehicles in each UAV's coverage area to complete various types of uploading tasks from the UAVs. To tackle the task scheduling issue, a deep reinforcement learning model is trained periodically, in order to reduce the task execution cost and the task completion time. Simulation was performed to investigate the performance of the proposed scheme from multiple aspects. A large number of experimental results prove that the scheme in this paper achieves both higher successful rate and lower task execution cost than baselines policies. In addition, the time of task assignment is short and the convergence of the results is guaranteed. In the future, task

offloading performance will be further improved and new applications based on parked vehicles will be studied.

Author Contributions: J.Z.: original draft writing, and review and editing; H.Z.: supervision; Y.W.: simulation; C.M. and Q.L.: review. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the Natural Science Foundation of China under grants (Nos. 61902282, 62002263, 61872083, 61872081) and in part by the Scientific Research Project of Tianjin Education Commission under Grant (No. 2021KJ186), and in part by the Science and technology project of Guangdong (No. 2020ZDZX3054).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Pu, C.; Zhou, X. Smartphone-Based Self Rescue System for Disaster Rescue. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019. [CrossRef]
2. Yin, S.; Zhao, Y.; Li, L. UAV-Assisted Cooperative Communications with Time-Sharing SWIPT. *IEEE Trans. Veh. Technol.* **2020**, *69*. [CrossRef]
3. Dridi, A.; Laroui, M.; Boucetta, C.; Afifi, H.; Moungla, H. Reinforcement Learning Vs ILP Optimization in IoT support of Drone assisted Cellular Networks. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 4589–4594.
4. Chen, Q.; Zhu, H.; Yang, L.; Chen, X.; Pollin, S.; Vinogradov, E. Edge computing assisted autonomous flight for uav: Synergies between vision and communications. *IEEE Commun. Mag.* **2011**, *59*, 28–33. [CrossRef]
5. Qiu, T.; Chi, J.; Zhou, X.; Ning, Z.; Atiquzzaman, M.; Wu, D.O. Edge computing in industrial internet of things: Architecture advances and challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2462–2488. [CrossRef]
6. Jiao, Y.; Wang, P.; Niyato, D.; Suankaewmanee, K. Auction mechanismsin cloud/fog computing resource allocation for public blockchain networks. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 1975–1989. [CrossRef]
7. Bai, T.; Wang, J.; Ren, Y.; Hanzo, L. Energy-efficient computation offloading for secure uav-edge-computing systems. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6074–6087. [CrossRef]
8. Chen, W.; Su, Z.; Xu, Q.; Luan, T.H.; Li, R. VFC-Based Cooperative UAV Computation Task Offloading for Post-disaster Rescue. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 1–9.
9. Young, C.; Man, L.; Koonjul, Y.; Nagowah, L. A low cost autonomous unmanned ground vehicle. *Future Comput. Inform. J.* **2018**, *3*, 304–320.
10. Zhou, F.; Wu, Y.; Hu, R.Q.; Qian, Y. Computation Rate Maximization in UAV-Enabled Wireless-Powered Mobile-Edge Computing Systems. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1927–1941. [CrossRef]
11. Yang, Z.; Pan, C.; Wang, K.; Shikh-Bahaei, M. Energy Efficient Resource Allocation in UAV-Enabled Mobile Edge Computing Networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 4576–4589. [CrossRef]
12. Zhang, J.; Zhou, L.; Tang, Q.; Ngai, E.C.H.; Hu, X.; Zhao, H.; Wei, J. Stochastic Computation Offloading and Trajectory Scheduling for UAV-Assisted Mobile Edge Computing. *IEEE Internet Things J.* **2019**, *6*, 3688–3699. [CrossRef]
13. 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.* **2017**, *67*, 2049–2063. [CrossRef]
14. Liu, Y.; Xiong, K.; Ni, Q.; Fan, P.; Letaief, K.B. UAV-Assisted Wireless Powered Cooperative Mobile Edge Computing: Joint Offloading, CPU Control, and Trajectory Optimization. *IEEE Internet Things J.* **2019**, *7*, 2777–2790. [CrossRef]
15. Hu, X.; Wong, K.K.; Yang, K.; Zheng, Z. UAV-Assisted Relaying and Edge Computing: Scheduling and Trajectory Optimization. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 4738–4752. [CrossRef]
16. Sun, H.; Zhang, B.; Zhang, X.; Yu, Y.; Sha, K.; Shi, W. FlexEdge: Dynamic Task Scheduling for a UAV-Based on-Demand Mobile Edge Server. *IEEE Internet Things J.* **2022**, *9*, 15983–16005. [CrossRef]
17. Zhang, T.; Xu, Y.; Loo, J.; Yang, D.; Xiao, L. Joint Computation and Communication Design for UAV-Assisted Mobile Edge Computing in IoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5505–5516. [CrossRef]
18. Garg, S.; Singh, A.; Batra, S.; Kumar, N.; Yang, L.T. UAV-Empowered Edge Computing Environment for Cyber-Threat Detection in Smart Vehicles. *IEEE Netw.* **2018**, *32*, 42–51. [CrossRef]
19. Han, R.; Wen, Y.; Bai, L. Rate Splitting on Mobile Edge Computing for UAV-aided IoT Systems. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *6*, 1193–1203. [CrossRef]

20. Alsenwi, M.; Tun, Y.K.; Pandey, S.R.; Ei, N.N.; Hong, C.S. UAV-Assisted Multi-Access Edge Computing System: An Energy-Efficient Resource Management Framework. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 January 2020. [CrossRef]
21. Xujie, L.I.; Zhou, L.; Sun, Y. Multi-task offloading scheme for UAV-Enabled Fog Computing networks. *EURASIP J. Wirel. Commun. Netw.* **2020**, *4*, 230. [CrossRef]
22. Zhan, C.; Hu, H.; Sui, X.; Liu, Z.; Niyato, D. Completion Time and Energy Optimization in the UAV-Enabled Mobile-Edge Computing System. *IEEE Internet Things J.* **2020**, *7*, 7808–7822. [CrossRef]
23. Chen, J.; Chen, S.; Luo, S.; Wang, Q.; Cao, B.; Li, X. An intelligent task offloading algorithm (iTOA) for UAV edge computing network. *Digit. Commun. Netw.* **2022**, *6*, 433–443. [CrossRef]
24. Wang, G.; Yu, X.; Xu, F.; Cai, J. Task offloading and resource allocation for UAV-assisted mobile edge computing with imperfect channel estimation over Rician fading channels. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 169–180. [CrossRef]
25. Zhou, Y.; Cheng, N.; Lu, N.; Shen, X.S. Multi-uav-aided networks:Aerial-ground cooperative vehicular networking architecture. *IEEE Veh. Technol. Mag.* **2015**, *10*, 36–44. [CrossRef]
26. Liu, N.; Liu, M.; Chen, G.; Cao, J. The The sharing at roadside: Vehicular content distribution using parked vehicles. In Proceedings of the 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 1–9.
27. Huang, X.; Yu, R.; Liu, J.; Shu, L. Parked Vehicle Edge Computing: Exploiting Opportunistic Resources for Distributed Mobile Applications. *IEEE Access* **2018**, *6*, 66649–66663. [CrossRef]
28. Ma, C.; Zhu, J.; Liu, M.; Zhao, H.; Liu, N.; Zou, X. Parking Edge Computing: Parked-Vehicle-Assisted Task Offloading for Urban VANETs. *IEEE Internet Things* **2022**, *8*, 9344–9358. [CrossRef]
29. Su, Z.; Hui, Y.; Guo, S. D2D-based content delivery with parked vehicles in vehicular social networks. *IEEE Wirel. Commun.* **2016**, *23*, 90–95. [CrossRef]
30. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-Efficient UAV Control for Effective and Fair Communication Coverage: A Deep Reinforcement Learning Approach. *IEEE J. Sel. Areas Commun.* **2022**, *36*, 2059–2070. [CrossRef]
31. Liu, M.; Gong, H.; Wen, Y.; Chen, G.; Cao, J. The last minute: Efficient Data Evacuation strategy for sensor networks in post-disaster applications. In Proceedings of the 2011 Proceedings IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 291–295.
32. Tang, C.; Wei, X.; Xiao, S.; Chen, W.; Fang, W.; Zhang, W.; Hao, M. A mobile cloud based scheduling strategy for industrial internet of things. *IEEE Access* **2018**, *6*, 7262–7275. [CrossRef]
33. Yan, J.; Bi, S.; Huang, L.; Zhang, Y.J.A. Deep reinforcement learning based offloading for mobile edge computing with general task graph. In Proceedings of the ICC 2020–2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7.
34. He, Y.; Zhai, D.; Jiang, Y.; Zhang, R. Relay Selection for UAV-Assisted Urban Vehicular Ad Hoc Networks. *IEEE Wirel. Commun. Lett.* **2020**, *9*, 1379–1383. [CrossRef]
35. Dynamic Programming. Available online: https://web.stanford.edu/class/cs97si/04-dynamic-programming.pdf (accessed on 13 May 2022).
36. Hu, J.; Ma, C.; Liu, N.; Liu, M.; Feng, W. VPOD: Virtual parking overlay network based data delivery in urban VANETs. *Int. J. Ad Hoc Ubiquitous Comput.* **2017**, *26*, 250–262.
37. Huang, W.; Guo, H.; Liu, J. Task Offloading in UAV Swarm-Based Edge Computing: Grouping and Role Division. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6.