UAV Frequency-based Crowdsensing Using Grouping Multi-agentDeep Reinforcement Learning

Cui ZHANG, En WANG, Funing YANG, Yong□jian YANG , Nan JIANG

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221100114

UAV Frequency-based Crowdsensing Using Grouping Multi-agent Deep Reinforcement Learning

Cui ZHANG[1], En WANG[1], Funing YANG[1], Yongjian YANG[1] and Nan JIANG[2]
1 College of Computer Science and Technology, Jilin University, Changchun 130012, China
2 College of Information Engineering, East China Jiaotong University, Nanchang 330013, China
(zhangcui20@mails.jlu.edu.cn)

Abstract Mobile CrowdSensing (MCS) is a promising sensing paradigm that recruits users to cooperatively perform sensing tasks. Recently, unmanned aerial vehicles (UAVs) as the powerful sensing devices are used to replace user participation and carry out some special tasks, such as epidemic monitoring and earthquakes rescue. In this paper, we focus on scheduling UAVs to sense the task Point-of-Interests (PoIs) with different frequency coverage requirements. To accomplish the sensing task, the scheduling strategy needs to consider the coverage requirement, geographic fairness and energy charging simultaneously. We consider the complex interaction among UAVs and propose a grouping multi-agent deep reinforcement learning approach (G-MADDPG) to schedule UAVs distributively. G-MADDPG groups all UAVs into some teams by a distance-based clustering algorithm (DCA), then it regards each team as an agent. In this way, G-MADDPG solves the problem that the training time of traditional MADDPG is too long to converge when the number of UAVs is large, and the trade-off between training time and result accuracy could be controlled flexibly by adjusting the number of teams. Extensive simulation results show that our scheduling strategy has better performance compared with three baselines and is flexible in balancing training time and result accuracy.

Keywords UAV Crowdsensing, Frequency coverage, Grouping multi-agent deep reinforcement learning
Chinese Library Classification TP393

1 Intorduction

With the increasing popularity of wearable devices, Mobile CrowdSensing (MCS) has recently become a promising sensing paradigm by recruiting users to perform various sensing tasks[1-2]. It has been widely used in environment monitoring[3], intelligent transportation[4], smart cities[5], etc. However, for some special tasks such as epidemic monitoring and earthquakes rescue, people with sensing devices could not enter the target task areas[6]. To this end, unmanned aerial vehicles (UAVs) as the powerful sensing devices equipped with different kinds of high-precision sensors could be used to replace user participation and carry out the special crowdsensing tasks[7-8].

Fig. 1 (left) shows an example of epidemic monitoring by UAV crowdsensing, there are 4 UAVs taking off from the same point (bottom right corner) and 3 charging stations in the map. UAVs are used to monitor the flow of people in some areas without sensors by frequently collecting data (e.g., photos) in these Point-of-Interests (PoIs). Different PoIs usually have different frequency coverage requirements. For example, because of different crowd intensity, a monitoring task requires 1 time per half an hour at the intersection, while 1 time per hour at the lakeside. Specially, 1 time per hour means sensing more than one times in a certain hour is regarded as only sensing once. Each UAV has its own flight path, and UAVs complete the PoI sensing task together. There are lots of similar scenarios in precision agriculture, environmental monitoring and so on.

In the scenario of Fig. 1, we need to design a scheduling algorithm for UAVs to meet the frequency coverage requirements as much as possible. Meanwhile, the fairness of different PoIs is also important, which means covering some PoIs too many times while leaving other PoIs uncovered is not suitable. In addition, because of the limited initial energy reserve, UAVs should keep balancing between data collection and energy charging. Hence, this problem turns into proposing a scheduling algorithm taking frequency coverage requirement, geographic fairness and energy charging into consideration simultaneously. Some existing approaches solve the similar scheduling problem by optimization theory, dynamic programming or centralized reinforcement learning (RL). Zhang et

Submission data:2022-11-14 Modification data:2023-01-03
This work was supported by the Innovation Capacity Construction Project of Jilin Development and Reform Commission(2020C017-2) and Science and Technology Development Plan Project of Jilin Province(20210201082GX).
Corresponding author:Funing YANG(yfn@jlu.edu.cn)

al.[9] scheduled mobile chargers to deliver energy to sensor nodes collaboratively by a dynamic programming algorithm "PushWait". Such approach could not be used to solve our problem because it needs to give definite pay-off function. Obviously, in our problem, many objectives need to be achieved simultaneously and the influence among UAVs is so complex that quantifying the effect when an UAV makes a decision is almost impossible. The centralized RL could not also be directly used in our problem because UAVs cannot communicate in real time for such a large target area, thus we cannot know the strategies of each UAV. Hence, how to schedule UAVs distributively under considering their interaction is the first challenge. To deal with this, we propose to formulate our problem based on multi-agent DDPG (i.e., MADDPG)[10] and regard each UAV as an agent. In this way, we could learn policies that require complex multi-agent coordination through the historical feedbacks. Then agents make decisions distributively according to the training results.

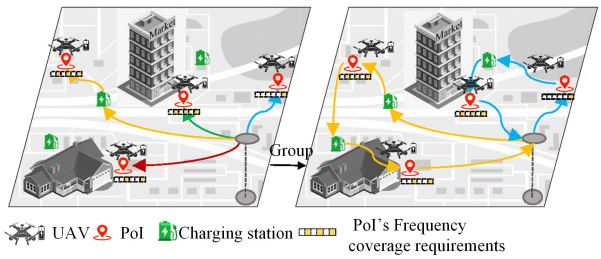


Fig. 1 UAV frequency-based crowdsensing

Some existing works use MADDPG to schedule UAVs but they concentrate on efficient data collection. In these works, there is a large amount of data in each PoI, once an UAV reaches a certain PoI, it can sense data unless the data of this PoI have been completely sensed[6,11-12]. However, in our problem, we focus on sensing PoIs with frequency coverage requirements. UAVs only can get an effective data when the PoI still needs to be sensed. Visiting one PoI frequently may not get more needed data, so a better scheduling strategy might be to schedule all UAVs to patrol the PoIs. Hence, the second challenge is how to re-adjust reward function so that MADDPG is suitable to solve this frequency coverage scheduling problem. To meet this challenge, we add a new function to the reward function, it returns an immediate reward by considering coverage time and other UAVs' actions.

Moreover, the number of agents in existing MADDPG works usually ranges from 1 to 5[11-12], research results show that a large number of agents make the training time too long to converge. In our patrol scheduling scenario, because of the large crowdsensing area and frequency coverage requirements, lots of UAVs are needed, so shortening training time is necessary. Mean field approximation is an approach to solve this issue[13], it shortens the training time fixedly and leads to a fixed loss in result accuracy meanwhile. However, the fixed reductions of training time and result accuracy may not satisfy the specific task demands. So a better way should control a trade-off between them according to task demands. Hence, the third challenge is how to balance the training time and result accuracy flexibly. To deal with this, we propose a grouping multi-agent deep reinforcement learning approach G-MADDPG. Specifically, a distance-based clustering algorithm DCA is proposed to group all UAVs into some "multi-agent teams". We use G-MADDPG to schedule the routes of each team instead of scheduling all UAVs. An equivalent transformation from UAVs to team makes sure that the coverage ability of each team is proportional to the number of UAVs in it. Then after determining the scheduling strategy of each team, each UAV patrols along the route of its team with a certain starting time difference. As shown in Fig. 1 (right), 4 UAVs are grouped into 2 teams. Each team has 2 UAVs and each UAV flies along the trajectory of their own team. In this way, G-MADDPG controls the trade-off between training time and result accuracy flexibly by adjusting the number of teams.

In summary, our main contributions are as follows:

(1) We prove that the problem of scheduling UAVs to sense PoIs with frequency coverage requirements is NP-hard, and formulate the problem based on multi-agent deep reinforcement learning model.

(2) We re-adjust the reward function of MADDPG by adding a new function which returns an immediate reward by considering coverage time and other UAVs' actions for dealing with the frequency coverage requirements.

(3) We propose a grouping approach G-MADDPG and a distance-based clustering algorithm DCA to balance the training time and result accuracy flexibly when the number of UAVs is large.

(4) We conduct extensive simulations, results show that our strategy performs well and can balance the trade-off between training time and result accuracy.

2 Related Work

2.1 Task Allocation

There are plenty of existing works that focus on task allocation in MCS. In [14], the authors proposed a personalized privacy-preserving task allocation framework for mobile crowdsensing that can provide personalized location privacy protection. Wang et al.[15] align task sequences with citizen's mobility and study the task allocation problem as a pattern matching problem. In addition, UAVs which equip different kinds of high-precision sensors have been used in more and more sensing tasks[6-8]. The work of [16] proposes a novel framework to control UAVs collecting most required data in the sensing region, but assumes only one UAV. Liu et al. con-

sidered a group of UAVs to provide data collection services[6,11]. Although some works study the problem of UAV crowdsensing, the scenarios they consider are different from our scenario where different PoIs have different frequency requirements.

2.2 Deep Reinforcement Learning

Recently, reinforcement learning has been considered quite successful to solve complex sequential decision-making problems, which can be described by a Markov Decision Process (MDP). Deep reinforcement learning (DRL) such as Deep Q-learning (DQN)[17], A3C[18] and so on[19-20], uses deep neural networks (DNNs) for model representation. Furthermore, multi-agent RL is proposed to deal with more dynamic and complex environments[21]. In the work of [22], the authors proposed a novel multi-agent RL framework (GCC-MARL) to solve the problem that leverages for-hire vehicles to collect city-scale sensory data. Lowe et al.[10] proposed the multi-agent deep deterministic policy gradient (MADDPG) for cooperative or competitive scenarios. Some existing works solve complex data collection problems by MADDPG[11-12]. However, these works pay attention to data collection volume not frequency coverage requirements, nor do they consider flexible control of the trade-off between training time and result accuracy when the training time is too long due to the large number of agents.

3 System Model and Problem Definiton

3.1 System Model

Without loss of generality, we consider that there is a set $U=\{u \mid u=1,2,\cdots,U\}$ of UAVs which can fly, sense data and charge themselves in a 2D target area. In our model, the target area is square and has a fixed border that UAVs cannot go beyond. Besides, the entire target area is divided into equal size square grids. We assume that there is a set of PoIs, denoted as $P=\{p \mid p=1,2,\cdots,P\}$ in the target area. These PoIs may have different frequency sensing requirements, denoted as $D=\{d \mid d=1,2,\cdots,D\}$. Let set $C=\{c \mid c=1,2,\cdots,C\}$ be C charging stations deployed in the area, each of them associates enough energy and can charge energy for each UAV. We assume that a data collection task lasts for some hours and it is discretized into equal length time steps, i.e., T time slots. In the beginning of the task, all UAVs are equipped with full energy e_0^u, $\forall u \in U$ at the same take-off location. Then at each time slot t, each UAV takes an action simultaneously. They decide to fly to a neighborhood or stay, sense data or charge energy. We assume that sensing data such as taking a photo would not take much time, it can accomplish in the current time slot. In addition, we define the UAV's sensing capacity as its sensing range R, i.e., for any PoI $p \in P$ in range R is considered to be collected, for any charging station $c \in C$ in range R is considered to be charged. We assume that UAVs cost some energy when they take an action, denoted as φ^u, $\forall u \in U$. Specifically, an UAV costs e^n energy for flying to neighborhood, e^s for staying in place and e^d for sending data. Apparently, the energy carried by UAVs decreases over time, so UAVs need to determine when and where to charge in time to avoid stopping working or crashing. We assume that a charging station can supply a portion of UAV's full energy in one time slot. If UAV is not full yet after charging one time slot, it can make a decision to go on charging in the next time slot or fly away, but each UAV cannot carry more energy than its capacity e_0^u.

3.2 Problem Definition

In our problem, when a task finishes, we calculate the sum times that all UAVs collect, which is denoted by D_T, as:

$$D_T=\sum_{p=1}^{P}\varphi(p) \tag{1}$$

where $\varphi(p)$ denotes the total collected times of PoI p. Then one of our objectives is to maximize D_T. However, only considering the collected times may lead to an unfair collection process which is not useful enough for our problem that monitors the flow of people during the epidemic. Therefore, we consider the geographical fairness of collected data among all PoIs, which is defined by the Jain's fairness index[23] as:

$$G_T=(\sum_{p=1}^{P}\psi(p))^2/P\sum_{p=1}^{P}\psi(p)^2 \tag{2}$$

$$\psi(p)=\varphi(p)/n_p \tag{3}$$

where n_p denotes the total times needed of PoI p, and $\psi(p)$ is the collection ratio of PoI p, i.e., the collected times of PoI p divides the total times needed of PoI p. Obviously, the closer the collection ratios of each PoI are, the higher geographical fairness is. Then another objective is to maximize G_T. Besides, we need to ensure that during the task, each UAV won't work abnormally due to lack of energy.

To sum up, we need to find a strategy π which can consider total collected times, geographical fairness and energy simultaneously, which is challenging. On the one hand, we need to consider frequency coverage requirements and interaction. In our problem, some PoIs may have higher requirements, so UAVs need to move back to sense them frequently. Besides, multiple UAVs complete the sensing task together, so each UAV's action needs to consider other UAVs' influence. On the other hand, we need to consider dynamical environment and charging. At each time slot, the entire environment changes itself because some PoIs may be sensed, UAVs may change their location and remaining energy. Moreover, for each UAV, when and where to charge is an issue. Charging at different time and different charging stations may influence whether UAVs miss the sensing demands of PoIs. Charging at an appropriate time can provide adequate support for future sensing. Similarly, going to different charging stations may make UAVs in different surroundings, which may impact the

total sensing times and the cooperation among all UAVs.

3.3 NP-hard Proof

Theorem 1 Our UAVs scheduling problem is NP-hard.

Proof. The main idea is to reduce the orienteering problem[24] which is NP-hard, to a special case of our problem. We consider a special case with the following restrictions. Let $|U|=1$ and $|C|=0$, that is there is only one UAV and no charging station. The UAV cannot charge during the task, therefore it only has the initial full energy that it carries at the beginning of the task. We also simplify that each PoI only needs to be sensed one time during the whole task. In this special case, our problem is to find a scheduling strategy for this UAV to make it fly around the environment, and the goal is to maximize the total sensing times only. Because when each PoI only needs to be sensed once, the geographical fairness is proportional to the total sensing times. In the orienteering problem, there is a set of vertices and each one has a score. The goal is to determine a path of limited length which visits some vertices to maximize the sum of collected scores. Therefore, the orienteering problem is reduced to a special case of our UAVs scheduling problem and our UAVs scheduling problem is NP-hard.

4 Problem Formulation

We model our problem as a Markov decision process (MDP), defined as a five tuple $\langle S, O, A, R, \gamma \rangle$.

(1) State: We denote that state has three parts, $S=\{s_t=(S_1,S_2,S_3)\}$. S_1 is the geographical information of PoIs and each PoI's frequency coverage requirements, i.e., $S_1=\{p_x,p_y,d_p\}_{p\in P}$, where p_x, p_y denote the coordinates of PoI p in the target area; S_2 is the UAVs position and their remaining energy, i.e., $S_2=\{u_{t_x},u_{t_y},e_t^u\}_{u\in U}$, where u_{t_x}, u_{t_y} denote the coordinates of UAV u at time slot t; S_3 is the collected times of each PoI at time slot t, denoted as $\varphi(p)_t$, i.e., if a PoI p is sensed at time slot $t+1$, the $\varphi(p)_{t+1}$ is calculated as $\varphi(p)_{t+1}=\varphi(p)_t+1$. S_3 is used to calculate collection ratio, then the collection ratio is used to calculate the geographical fairness among all PoIs. The geographical fairness is to ensure the data diversity.

(2) Observation: The observation of each UAV includes its own position, remaining energy and the condition of its eight neighbors. Specifically, the condition of its neighbors includes whether there is a PoI that needs to be collected, whether there is a charging station, whether there are other UAVs, whether the neighbor is the fixed border, whether there are both PoIs that needs to collected and other UAVs and so on. The observation can be denoted as $o_t^u=\{(u_{t_x},u_{t_y},e_t^u,o_{t0}^u,o_{t1}^u,\cdots,o_{t8}^u)\}$, where o_{ti}^u denotes the condition of UAV's own position and its neighbors, so the number of o_n^u is nine.

(3) Action: We denote the action as $A=\{a_t=(a_t^1,a_t^2,\cdots,a_t^u)\}$. At each time slot t, each UAV takes an action a_t^u that indicates whether it stays in the current grid or moves to one of eight neighbors. If it reaches a grid where there is a PoI which needs to be sensed, the UAV will sense the data. If it reaches a grid where there are charging stations, it will charge some energy. All a_t^u compose a_t.

(4) Reward: At each time slot t, UAVs take their own action a_t^u, then the whole state of the environment will change and each UAV u will obtain the reward r_t^u:

$$r_t^u=f_t\delta_t^u-\rho_t^u \tag{4}$$

where f_t represents the geographical fairness calculated by (2). δ_t^u is a new function added for focusing on frequency coverage requirements, which represents whether an UAV u collects data successfully, so $\delta_t^u\in\{0,1\}$. δ_t^u depends on two points. The first is whether there is a PoI that is still needed to sense at this time slot in the UAV's position. Another one is whether there are other UAVs that try to sense the data in this position. ρ_t^u represents the penalty that UAV u receives when it hits the fixed border or it is close to running out of energy. Specifically, the penalties in the two cases are different.

(5) Discount factor γ: It shows the importance of future reward compared to present reward and $0<\gamma<1$.

Problem Formulation: At each time slot t, every UAV determines its action based on its observation and receives corresponding reward from the environment. The state of whole environment will change over time. For each UAV, the problem can be formulated as:

$$V_t^u(O_t^u)=\max_{a_t^u}[r_t^u(O_t^u,a_t^u)+\gamma V_{t+1}^u(O_{t+1}^u)] \tag{5}$$

Therefore, the optimal strategy of each UAV is given by:

$$\pi^u=\arg\max_{a_t^u}[r_t^u(O_t^u,a_t^u)+\gamma V_{t+1}^u(O_{t+1}^u)] \tag{6}$$

Obviously, our problem cannot be solved by traditional optimization method, because when an UAV takes an action, the future reward from this action cannot be calculated precisely. Besides, our scenario is a multiple UAVs environment, there is influence among UAVs, an UAV's reward is affected by others. Difficult real-time communication makes centralized RL infeasible. In this way, we opt to solve our problem based on MADDPG. Moreover, because of large agents, balancing the training time and result accuracy flexibly is challenging.

5 Scheduling Strategy: G-MADDPG

5.1 Overview

In this section, we propose our scheduling strategy G-MADDPG, which is a grouping multi-agent deep RL approach that groups all UAVs into some teams. First of all, we assume that we have determined the number of teams and the number of UAVs in each team, then we regard each team as an agent, discuss the system flow of the teams interacting with

environment and determine teams scheduling strategy. Next, we focus on the details of GMADDPG, including a time interval flying pattern for UAVs and a distanced-based clustering algorithm DCA. This flying pattern for UAVs ensures an equivalent result as team strategy and DCA can group UAVs into teams and adjust the number of teams. Finally, we summarize the flow of G-MADDPG.

5.2 Scheduling of Multi-agent Team

We assume that UAVs have been grouped into a set of teams, denoted as $V=\{v \mid v=1,2,\cdots,V\}$. Then we determine team scheduling strategy based on multi-agent deep deterministic policy gradient method; we regard each team as an agent and the whole framework is shown as Fig. 2.

(1) Taking Action after Observation: At the beginning of the time slot t, each team gets its own observation o_t^v from the environment. o_t^v is the same as the form of UAVs observation, so $o_t^v=\{v_{t_x}, v_{t_y}, e_t^v, o_{t0}^v, o_{t1}^v, \cdots, o_{t8}^v\}$. Then based on this observation, each team decides an action a_t^v to take by its actor network. This process happens on all teams and all a_t^v compose a_t. Then they execute their own action.

(2) Storing Transitions: After all teams have performed actions, the environment returns corresponding reward r_t^v to each team calculated by (4). Next, the environment updates the state, including each PoI's collection $\varphi(p)_t$, teams' location (v_{t_x}, v_{t_y}) and their remaining energy e_t^v. Finally, each team v is in a new state s_{t+1} and gets a new observation o_{t+1}^v. Hence, we can get a four tuple $\langle o_t, a_t, r_t, o_{t+1}\rangle$ for every time slot as shown in the bottom right corner of Fig. 2, where o_t, o_{t+1}, a_t, r_t are consisted of all teams' observations, actions and rewards correspondingly. In addition, these tuples at different time slots are stored in a replay buffer for training.

(3) Training Process: As we can see from Fig. 2, each team is implemented with four DNNs, which are actor network $\pi^v(o_t^v)$, critic network $Q^v(o_t^v, a_t)$ and their target networks $\pi'^v(o_t^v)$, $Q'^v(o_t^v, a_t)$. The actor network $\pi^v(o_t^v)$ is used to determine an action for team at each time slot. The critic network $Q^v(o_t^v, a_t)$ is used to estimate the reward of different observations and actions. For example, after a team v has observed the environment and performed its action, the next time this team obtains the same observation, if the team executes a different action and gets a bigger reward, then the actor network should increase the probability of taking the new action. Target networks $Q'^v(o_t^v, a_t)$ and $\pi'^v(o_t^v)$ are used to help stabilize learning. Their parameters are periodically "soft" updated with the most recent parameters of $Q^v(o_t^v, a_t)$ and $\pi^v(o_t^v)$. Hence, we propose the method to update networks. During training, we first sample a mini-batch of N transitions from replay buffer. For each team v, its target actor network $\pi'^v(o_t^v)$ gives a target action a'^v_{t+1} with given observation o_{t+1}^v from the mini-batch. Then the critic network Q^v is updated by minimizing a loss function $L(\theta^{Q^v})$ as:

$$L(\theta^{Q^v})=\mathbb{E}[Q^v(o_t^v, a_t^1, a_t^2, \cdots, a_t^V \mid \theta^{Q^v})-y_t^v]^2 \tag{7}$$

where θ^{Q^v} denotes the parameters of the critic network, and the first part Q^v is the Q-value with observations and actions from the replay buffer, and the second part y_t^v is the target Q-value which is computed by:

$$y_t^v=r_t^v+\gamma Q'^v(o_{t+1}^v, a_{t+1}'^1, a_{t+1}'^2, \cdots, a_{t+1}'^V \mid \theta^{Q'^v}) \tag{8}$$

Obviously, the key of updating the critic network is to reduce the difference between the predicted reward and real reward, which is to improve the accuracy of prediction. Note that when predicting the reward (i.e., the target Q-value), both the immediate reward r_t^v and the possible future reward $Q^{v\prime}$ are taken into consideration, and the parameter γ is used to balance the importance of possible future reward.

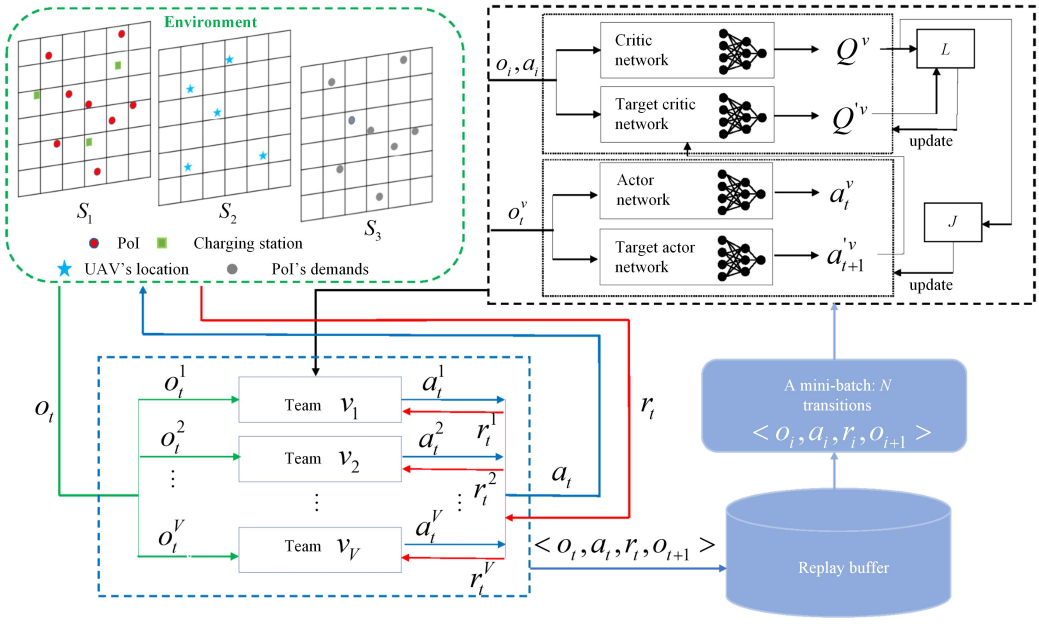


Fig. 2 Team scheduling framework

Next, π^v is updated by using gradients as:

$$\nabla_{\theta^{\pi^v}} J = \mathbb{E}[\nabla_{\theta^{\pi^v}} \pi^v(o|\theta^{\pi^v})|_{o=o_t^v} \cdot \nabla_a Q^v(o, a_t^1, a_t^2, \cdots, a_t^V| \theta^{Q^v})|_{a_t^v=\pi^v(o_t^v)}] \quad (9)$$

where θ^{π^v} denotes the parameters of the actor network. Finally, the two target networks are updated softly as:

$$\theta^{Q'^v} = \tau\theta^{Q^v} + (1-\tau)\theta^{Q'^v} \quad (10)$$

$$\theta^{\pi'^v} = \tau\theta^{\pi^v} + (1-\tau)\theta^{\pi'^v} \quad (11)$$

where $0 < \tau < 1$.

(4) Testing Process: After training the DNNs for each team, the model (i.e., the parameters in four networks) is saved for testing. During testing, we only need the trained actor network $\pi^v(o_t^v)$. It produces an action a_t^v by going through the DNN of weights θ^{π^v}, when given team v's own observation o_t^v. Then after taking this action, the environment gives a reward r_t^v and changes v's observation to o_{t+1}^v, and team v continues to determine next action. Therefore, our algorithm does not need any communication among teams during execution phase. Each team can make its decision by its own observation which reflects that our G-MADDPG is a distributed strategy.

The detailed algorithm is shown in Algorithm 1 which includes the training process and environment interactions. At the beginning, for each team v, we initialize its critic network Q^v and actor network π^v randomly with weight θ^{Q^v} and θ^{π^v} respectively (line 2). Two target networks with parameters $\theta^{Q'^v}, \theta^{\pi'^v}$ are copied from their critic and actor networks (line 3). Then we initialize replay buffer B which is used to store transitions (line 4). In each episode, we initialize environment and obtain the initial state s_0. Then the data sensing which lasts for T time slots starts. At the beginning of time slot t, each team selects an action $a_t^v = \pi_{\theta^{\pi^v}}(o_t^v)$ and adds a noise for exploration (line 9). Then environment executes all actions and gives a set of rewards r_t to teams (lines 10-11). Next, transition $\langle o_t, a_t, r_t, o_{t+1} \rangle$ is stored to replay buffer B, if it is full, the oldest one will be dropped out (lines 12-14). If the transitions in replay buffer are enough for training, we sample N transitions and update critic network by minimizing the loss function, actor network by using gradients. Two target networks are softly updated correspondingly (lines 15-17). After adequate training, the model is saved for testing.

5.3 Grouping Multi-agent Deep RL Approach

As the number of UAVs increases, the interaction between UAVs and environment grows simultaneously, moreover, the interaction among UAVs grows exponentially. These two points cause the training convergence speed to be very slow. In order to solve this, we propose a grouping multi-agent DRL method G-MADDPG. Compared with mean field approximation which makes a fixed trade-off between training time and result accuracy, our method can control the trade-off flexibly according to specific task demands. In practice, some problems focus more on result accuracy, while others focus more on training time, so making a flexible trade-off is needed.

G-MADDPG controls this trade-off flexibly by grouping all UAVs into different number of teams. It regards each team as an agent and considers the interaction among teams. If we group all UAVs into more teams, we will consider the interaction more accurately, then the result accuracy will be higher and the training time will be longer. Specially, if we don't group, we will consider the most number of agents, then the result will be the best and the training time will be the longest. If we group all UAVs into one team, we will consider only one agent, then the training time will be the shortest and the result will be the worst.

For details of G-MADDPG, firstly, we cluster all UAVs into several teams dynamically by considering specific task requirements and scenarios. We call each team "multi-agent team", and each team has a bigger ability, their flying speed is proportional to the number of UAVs in it. The purpose of this is to be able to obtain an equivalent UAV flight strategy after determining the team's strategy. Specifically, see below equivalent transformation. Then we determine team scheduling strategy by Algorithm 1. Finally, each UAV patrols at its original speed along the route of its team with a certain time difference.

Algorithm 1 Teams Scheduling Strategy

Input: A set of teams V, a set of PoIs P with frequency coverage requirements D, a set of charging stations C, discount factor γ, update rate τ

Output: Actor network, critic network, target actor network, target critic network

1. for all team v=1,2,…,V do
2. Initialize critic network Q^v and actor network π^v randomly with weight θ^{Q^v} and θ^{π^v}
3. Initialize target network $Q'^v = Q^v$, $\pi'^v = \pi^v$
4. Initialize replay buffer B
5. for all episode=0,1,…,M-1 do
6. Initialize the environment, receive initial state s_0
7. for all time slot t=0,1,…,T-1 do
8. for all team v=1,2,…,V do
9. Get current observation o_t^v from the environ-ment, select an action $a_t^v = \pi_{\theta^{\pi^v}}(o_t^v)$ by the current actor network and add a noise for exploration
10. Execute $a_t = (a_t^1, a_t^2, \cdots, a_t^V)$ in environment
11. Environment changes state, teams obtain rew-ard r_t and next observation o_{t+1}
12. Store transition $\langle o_t, a_t, r_t, o_{t+1} \rangle$ to B

13. if B is full then then
14. Remove the oldest experience from B
15. if size(B)≥H then
16. Sample a mini-batch of N transitions from B
17. Update four networks by Algorithm 2.

Algorithm 2 Update Four Networks

1. Compute the target Q-value in mini-batch:
2. $y_t^v = r_t^v + \gamma Q'^v(o_{t+1}^v, a'^1_{t+1}, a'^2_{t+1}, \cdots, a'^V_{t+1} | \theta^{Q'^v})$
3. Update critic network θ^{Q^v} by minimizing the loss:
4. $L(\theta^{Q^v}) = \mathbb{E}[Q^v(o_t^v, a_t^1, a_t^2, \cdots, a_t^V | \theta^{Q^v}) - y_t^v]^2$
5. Update actor network θ^{π^v} using gradients:
6. $\nabla_{\theta^{\pi^v}} J = \mathbb{E}[\nabla_{\theta^{\pi^v}} \pi^v(o | \theta^{\pi^v})|_{o=o_t^v} \cdot$
7. $\nabla_a Q^v(o, a_t^1, a_t^2, \cdots, a_t^V | \theta^{Q^v})|_{a_t^v = \pi^v(o_t^v)}]$
8. Softly update the parameters of target networks by:
9. $\theta^{Q'^v} = \tau\theta^{Q^v} + (1-\tau)\theta^{Q'^v}$
10. $\theta^{\pi'^v} = \tau\theta^{\pi^v} + (1-\tau)\theta^{\pi'^v}$

The time difference is the ratio of the time that an UAV patrols once and the number of UAVs in this team. Fig. 3 shows an example that a team consists of three UAVs, left part is team's flying pattern and right is UAV's flying pattern. This interval flying pattern for UAVs makes the same result as team strategy, which can be proved as follows:

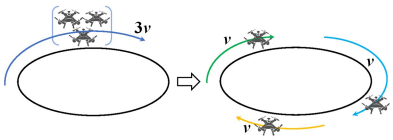


Fig. 3 Team strategy vs. UAVs strategy

The main idea is proving that the PoIs on the route is visited by the same time interval under the two patterns. We assume that the length of team's flight path under the scheduling strategy is s_l, the speed of UAV is v_{indi} and the number of UAVs in the team is n_{team}, so the speed of the team is $v_{\text{team}} = v_{\text{indi}} * n_{\text{team}}$. When the team patrols its route, the PoI which is on the route is arrived once per $t_{\text{team}} = s_l / v_{\text{team}}$ time slots. Correspondingly, when UAVs patrol along the route simultaneously, each UAV arrives the PoI once per $t_{\text{indi}} = s_l / v_{\text{indi}}$ time slots. However, PoIs are visited by all UAVs in the team, so the time interval is:

$$\frac{t_{\text{indi}}}{n_{\text{team}}} = \frac{s_l / v_{\text{indi}}}{n_{\text{team}}} = \frac{s_l / (v_{\text{team}} / n_{\text{team}})}{n_{\text{team}}} = \frac{s_l}{v_{\text{team}}} \tag{12}$$

which is the same as their team, so these two patterns make an equivalent result. Taking Fig. 3 as an example, we assume the path length is 6 and UAV's speed is 1, so team's speed is 3. For team strategy, it visits each PoI on the route once per 6/3=2 time slots, and for UAVs strategy, each PoI is visited once per (6/1)/3=2 time slots. These two results are equal. Similarly, the energy consumption and charging conditions of the two methods are also the same.

Another key for G-MADDPG is how to cluster all UAVs into several teams according to the specific task and scenarios. We propose a distance-based clustering algorithm DCA to solve this. Firstly, we cluster charging stations with the distance threshold ξ. Specifically, if the distance of two stations is less than ξ, these two stations are combined as a new element, and the location of this new element is the average of these two stations. Then the new element taking place of the original two stations continues to cluster with others. This process repeats until the distance of two closest elements is no less than ξ. After this, there are k_c station sets, and each set has a center location that is the average of all stations in it.

Secondly, we allocate all PoIs into these k_c sets. If the distance between one PoI and one set's center location is less than ξ, this PoI is allocated to this set. If one PoI is near several sets, it is allocated to the nearest one. Then each set's location is updated by the average of all stations and PoIs in it. This process is repeated until all PoIs are allocated or the nearest distance between remaining PoIs and sets is more than ξ. For remaining PoIs, we cluster them as the way clustering charging stations, then there will be k_p PoIs sets. Hence, the sum number of sets is $k = k_c + k_p$, which is also the number of teams. For each set, we calculate the sum requirements of PoIs in it, then assign UAVs proportionally. For example, there are 20 UAVs, they are grouped into four teams and each team needs 50, 100, 200 and 150 times to collect respectively, then the assigned UAV number of each team is 2, 4, 8 and 6.

Therefore, UAVs are grouped into k teams and each team has several UAVs. G-MADDPG determines teams scheduling strategy by Algorithm 1 first, then compares the training time and result accuracy with the task demand. If the training time is longer, it groups the two nearest teams into one team. If the result accuracy is smaller, it divides the with the largest team which has the most UAVs into two same size teams. This process is repeated until the task's demands are satisfied.

G-MADDPG is shown in Algorithm 3. It trains the teams that are clustered by DCA at first (lines 1—2), then adjusts the number of teams dynamically according to specific task demands until the demands are met (lines 3—8). For example, in a specific problem, UAVs are divided into four teams with 2, 6, 3, 4 UAVs in each team separately but the result accuracy does not meet the demand, G-MADDPG would divide the teams with 6 UAVs in two teams by distance-based clustering algorithm. Finally, after determining team strategy, UAVs patrol among their team route with a time interval difference (line 9).

Algorithm 3 Grouping Scheduling Strategy: G-MADDPG

Input: The demands of training time T_{train} and result accuracy R_{acc}

Output: UAVs scheduling strategy

1. Get the clustering result that UAV's team set S_U by DCA
2. Regard each team in the set S_U as an agent, train and get teams scheduling strategy by Algorithm 1
3. while training time t_{train} or result accuracy r_{acc} dissatisfies the demands do
4. if training time $t_{train} > T_{train}$ then
5. Group the nearest two teams into one team
6. if result accuracy $r_{acc} < R_{acc}$ then
7. Divide the team with the largest number of UAVs into two same size teams
8. Train and get teams scheduling strategy by Algorithm 1 under the new set S_U
9. UAVs patrol their team route with a time interval

In our problem, UAVs needs to patrol among PoIs and charging stations for meeting frequency coverage requirements and charging, so after determining teams' scheduling strategy, we can propose an equal UAV flying pattern that each UAV patrols among their team route with a time difference. However, our equal flying pattern does not suit all problems, for example, the problem which concentrates on efficient data collection[6]. In this problem, if UAVs patrol among their team route with a time difference, we won't get the same result as team strategy in data collection ratio and fairness.

6 Performance Evaluation

6.1 Setting

In our simulations, we set the target area as a 2D square. Each UAV starts with 60 units of energy reserve (as full energy). An UAV costs 10 units of energy when it flies to neighborhood, 5 units of energy for staying in place and 0.5 units of energy for sending data. We set penalty $\rho_v^t = 0.2$ for border collisions and $\rho_v^t = 2.0$ when UAV is close to running out of energy. UAV's sensing capacity is one square grid. We use PyTorch 1.0.1 to implement G-MADDPG, all experiments run on a server with GTX 1080Ti GPU cards, AMD Ryzen 7 2700 Eight-Core Processor CPU cards and 16 GB memory. Besides, in implementation, we set the learning rate as 0.001, discount rate factor $\gamma = 0.95$, buffer size as 1.6×104, batch size as 512 and soft update factor τ to 0.01.

6.2 Illustrative Moving Trajectories

We test our strategy and observe the results in three-agent and four-agent scenario. We observe that UAVs learn to cooperate in both scenarios. They are responsible for a subarea and patrol around the PoIs without going beyond the border. This reflects the penalty in reward function works. Also, UAVs fly to every PoI rather than moving among some PoIs which have higher frequency coverage requirements, because our strategy considers fairness simultane-ously. Besides, UAVs successfully learn to go to charging stations for charging and go on sensing data again. Fig. 4 shows the visits of three PoIs in the three-agent scenario, we can see although these PoIs have quite different requirements which are 1 time per 5, 1 time per 20 and 1 time per 8 time slots, UAVs almost cover all sensing demands, only missing one demand of PoI 1 at the first 5 time slots.

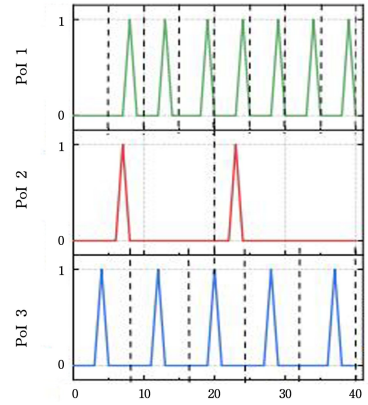


Fig. 4 Illustrative UAVs flying trajectories

6.3 Baseline Methods and Evaluation Metrics

There are three baselines as follows:

(1) IQL-DiffRewards[21]: Each UAV determines its actions by deep Q-learning and regards other agents as part of the environment. It does not consider other agents' strategies.

(2) NEC: UAVs visit the PoI which is the nearest and most needs to be sensed, besides, we assume that UAVs communicate with nearby UAVs to avoid choosing the same PoI.

(3) Random: Each UAV randomly selects their action.

We use the following metrics to measure the performance:

Frequency coverage ratio (F_T): $F_T = D_T / \sum_{p=1}^{P} n_p$, it calculates as a ratio between the total collected times and initial needed.

Geographical fairness (G_T): calculated by (2): $G_T = (\sum_{p=1}^{P} \psi(p))^2 / P \sum_{p=1}^{P} \psi(p)^2$ to show geographically how evenly each PoI's requirement is met.

Sum charging energy (C_T): $C_T = E_T / e_0^u$, it is the ratio between UAVs' total charging energy E_T during the task and the initial energy of UAV.

During testing, all algorithms except NEC repeat 10 times and we take the average of ten testing results for evaluation. NEC is tested only once since it produces the fixed results.

6.4 Comparing with Baselines

We conduct seven sets of simulations to evaluate our strategy. For testing, we generate a three agents scenario ran-

domly which includes 10 PoIs with random frequency coverage requirements and locations, and 4 charging stations with random locations. The target area is divided into a 9×9 grid-shape. We set the total time slots to 400, the charging proportion to 1 which means an UAV can get full energy in one time slot. We show the simulation results in Fig. 5—Fig. 12.

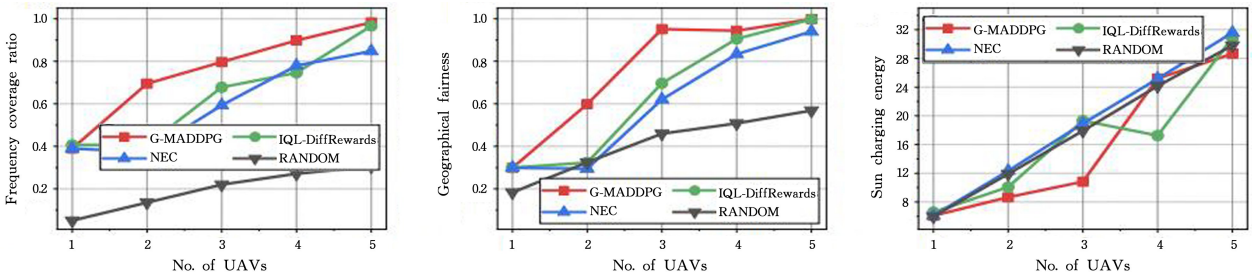


Fig. 5 Performances on different numbers of UAVs

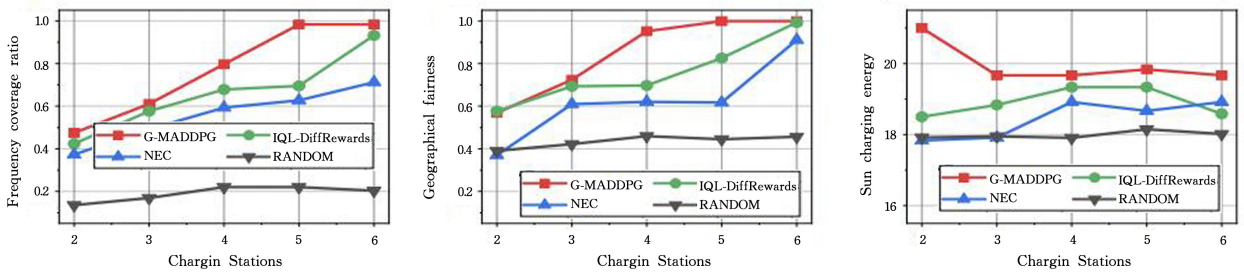


Fig. 6 Performances on different numbers of charging stations

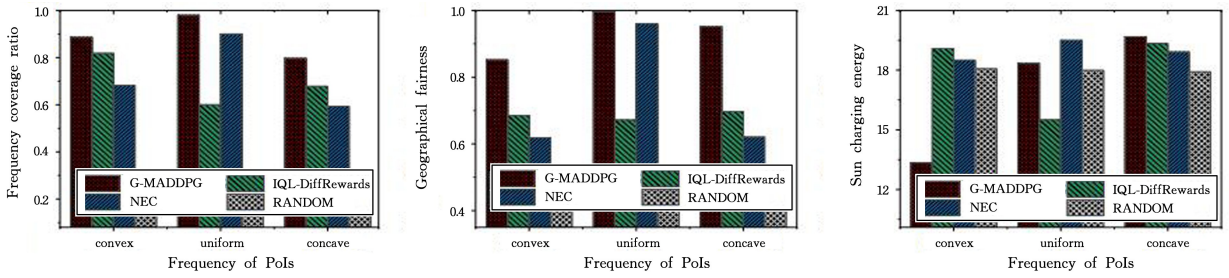


Fig. 7 Performances on different PoIs frequency requirements

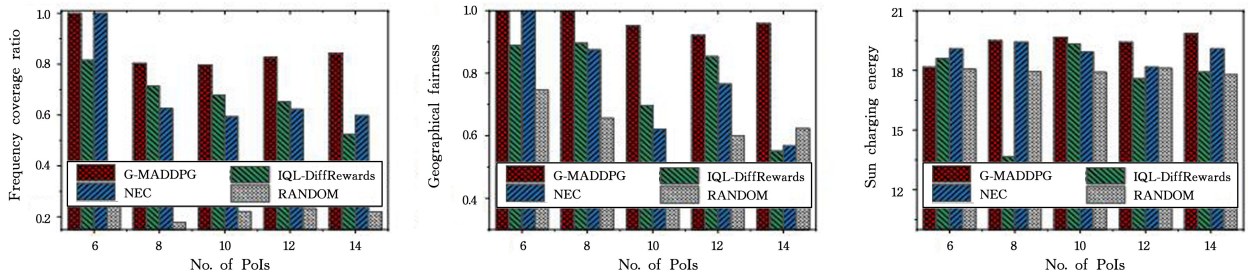


Fig. 8 Performances on different numbers of PoIs

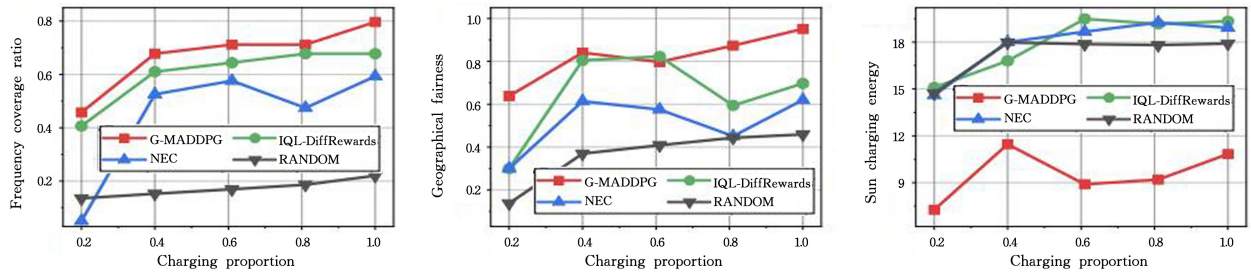


Fig. 9 Performances on different charging proportions

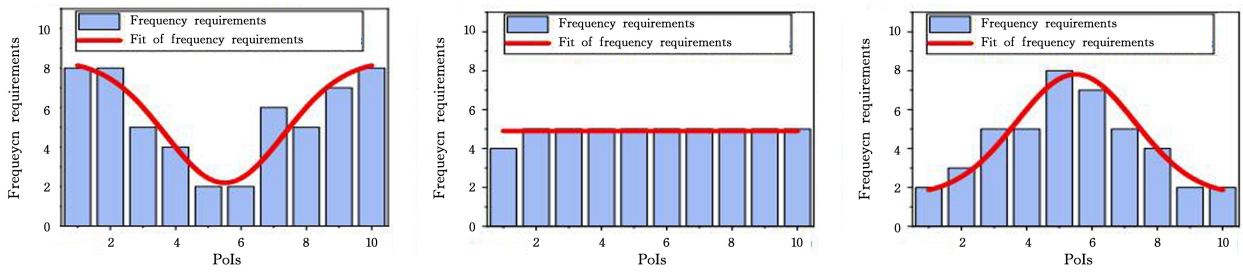


Fig. 10 Convex vs. uniform vs. concave distributions

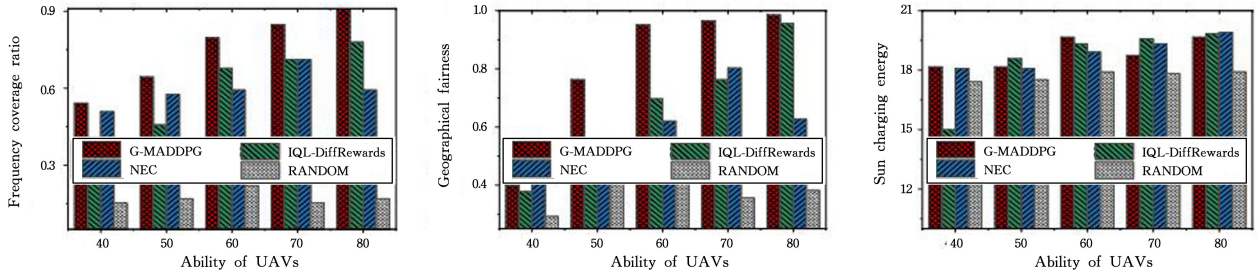
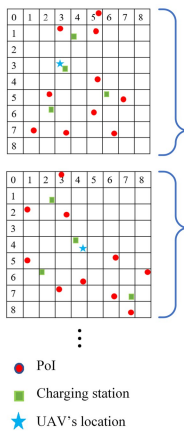


Fig. 11 Performances on different ability of UAVs



	PoIs locations	CS locations	UAVs taking off location	Frequency Coverage Ratio				Geographical Fairness			
				GM	IQL	NEC	RA	GM	IQL	NEC	RA
1	[(0,5), … (7,6)]	[(3, 3), … (5, 6)]	(3, 3)	**0.797**	0.678	0.593	0.22	**0.951**	0.696	0.62	0.46
2	[(0,3), … (8,7)]	[(1, 2), … (7, 7)]	(4, 4)	**0.797**	0.729	0.661	0.17	**0.881**	0.727	0.776	0.396
3	[(1,3), … (7,7)]	[(2, 4), … (6, 2)]	(5, 6)	**0.712**	0.627	0.543	0.203	**0.817**	0.7	0.597	0.48
4	[(2,4), … (8,5)]	[(3, 4), … (7, 5)]	(5, 2)	**0.745**	0.644	0.576	0.203	**0.828**	0.756	0.541	0.393

Fig. 12 Performances on different locations

(1) Performances on Different Numbers of UAVs: We first show the results when we change the number of UAVs from 1 to 5 and keep others fixed in Fig. 5. We can see G-MADDPG gets the best result in terms of F_T and G_T. Both F_T and G_T increase with more UAVs under all four strategies. This is because more UAVs have better ability to sense so they can miss less data and visit more PoIs. Furthermore, when the number of UAVs is one, there is no interaction between UAVs, so all strategies except Random get the same result. Then as the number is more than 1, the interaction appears, so neither IQL-DiffRewards which does not consider the interaction, nor NEC which considers the interaction simply by avoiding UAVs going to the same PoI can get a better result. From Fig. 5 (c), we can observe that although G-MADDPG gets a better result, C_T is almost less or equal to others, which means our strategy does not consume more energy for better performances.

(2) Performances on Different Numbers of Charging Stations: We change the number of charging stations from 2 to 6. From Fig. 6 (a, b), we can observe G-MADDPG consistently outperforms three baselines. Both F_T and G_T increase with numbers. Because of more charging stations, teams do not need to travel a long journey to charge energy, they can focus more on frequency coverage. Furthermore, our strategy learns how to better use the charging stations, thus both G_T and F_T do not increase and get 1.0 after $C=5$. From Fig. 6 (c), we can observe that G-MADDPG charges more energy for meeting more demands but the increments are quite small.

(3) Performances on Different PoIs Frequency Coverage Requirements: We conduct simulations to test performances of three different PoIs frequency coverage requirements distributions which are convex, uniform and concave as shown in Fig. 10. Specifically, PoIs frequency requirements in convex distribution means most of PoIs have higher requirements and a few UAVs are medium or lower frequency requirements. On the contrary, concave distribution means most of PoIs have

lower frequency requirements and a few UAVs are medium or higher requirements. Uniform distribution means all PoIs have similar frequency coverage requirements. From the results in Fig. 7, we can see that G-MADDPG gets the best performance, this is because whatever PoIs distribution follows, G-MADDPG can consider the interaction among teams well and schedule teams to meet different requirements cooperatively. From Fig. 7(c), we can draw the same conclusion as Fig. 5 (c) that our strategy does not get better results by consuming too much energy.

(4) Performances on Different Numbers of PoIs: More PoIs make UAVs explore more area to ensure frequency coverage ratio and geographical fairness. Meanwhile, the interaction among UAVs increases and the problem is more complex. Hence, we set 6 to 14 PoIs and fix others to test. As the results shown in Fig. 8, G-MADDPG outperforms all three baselines in F_T and G_T, and does not charge too much energy for better results. This reflects it can consider complex interaction and schedule UAVs well when the number of PoIs increases.

(5) Performances on Different Charging Proportions: We change charging proportion of full energy from 0.2 to 1. From the results in Fig. 9, G-MADDPG always obtains the maximal F_T and F_T gets higher as charging proportion increases. This is because the bigger charging proportion makes UAVs use more time slots to satisfy the task demands instead of ensuring flying normally. For G_T, G-MADDPG obtains the best result except charging proportion is 0.6. However, in this case, GMADDPG has a bigger improvement in F_T with a quite small reduction in G_T, so the comprehensive result is better than others. From Fig. 9 (c), we can see G-MADDPG consumes the least energy in all cases which reflects that our strategy can utilize the charging ability well and schedule UAVs efficiently.

Table 1 Result of three strategies

Strategies	G-MA (6)	MFA	G-MA (3)
Frequency Coverage Ratio	0.860	0.828	0.796
Geographical Fairness	0.927	0.891	0.840
Training Time/h	109.8	43.75	37.9

(6) Performances on Different Ability of UAVs to Carry Energy: All the above simulations focus on different scenarios and requirements, then we consider the impact of different UAV capabilities. We change the ability of UAVs to carry energy from 40 to 80 units. UAVs has higher ability to carry energy means they can fly farther and do not need to charge frequently. As shown in Fig. 11, G-MADDPG outperforms other strategies in F_T and G_T, and does not charge too much energy for better results. Both F_T and G_T increase as the ability of UAVs increases, because UAVs have more time slots to sense, they can miss less data and visit more PoIs.

(7) Performances on Different Locations of PoIs, Charging Stations and UAVs Taking off: Different locations of PoIs and UAVs taking off can influence cooperation between UAVs, different locations of charging stations affect the choice of charging time and location of UAVs. Hence, we generate four different scenarios randomly with only differences in locations in Fig. 12. We can observe that G-MADDPG gets the best performance in all four scenarios. This is because G-MADDPG can consider the interaction among teams in different scenarios and schedule teams to meet requirements cooperatively, while others cannot.

6.5 Flexible Trade-off

We conduct simulations to see the performance among G-MADDPG with different number of teams and mean field approximation. We set 6 UAVs, 18 PoIs and 4 charging stations in the target area. The results show that all strategies converge and the training time can be ranked as: G-MADDPG (3) (i.e., all UAVs are grouped into 3 teams) < Mean Field approximation < G-MADDPG (6) (i.e., all UAVs are grouped into 6 teams). Table. 1 shows the collection result, we can observe that G-MADDPG (6) gets the best results in F_T and G_T, mean field approximation and G-MADDPG (3) are similar but G-MADDPG (3) needs less training time. The training time of G-MADDPG (6) is nearly 3 times of G-MADDPG (3). Then, we adjust the number of teams based on DCA. As shown in Fig. 13, both the result accuracy and training time increase as the number of teams increases. Hence, our approach G-MADDPG can control the trade-off between training time and result accuracy flexibly by adjusting the number of teams.

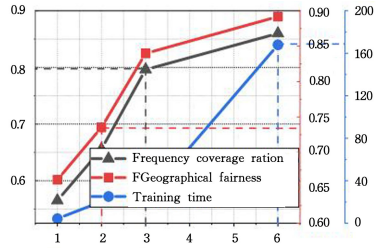


Fig. 13 Performance of G-MADDPG

CONCLUSION In this paper, we consider the problem of scheduling UAVs to sense PoIs which have different frequency coverage requirements. First, we prove that our scheduling problem is NP-hard and formulate the problem based on multi-agent deep reinforcement learning for considering the influence among UAVs. Then we re-adjust the reward function for focusing on frequency coverage requirements. In order to deal with that the training time is too long when the number of UAVs is large, we propose a grouping approach G-MADDPG based on a distance-based clustering algorithm DCA. It groups all UAVs into several teams and schedules teams distributively by considering the interaction among

teams. It can control the trade-off between the training time and result accuracy flexibly by dynamically adjusting the number of teams. Finally, we conduct extensive experiments, the results justify the performance of our approach and show that G-MADDPG is flexible to control the trade-off and that the result accuracy gets higher as the number of teams gets bigger.

REFERENCES

[1] YANG S, HAN K, ZHENG Z, et al. Towards personalized task matching in mobile crowdsensing via fine-grained user profiling [C]// IEEE INFOCOM. 2018: 2411-2419.

[2] WANG Z B, PANG X Y, CHEN Y H, et al. Privacy-Preserving Crowd-Sourced Statistical Data Publish-ing with An Untrusted Server [J]. IEEE Transactions on Mobile Computing, 2019, 18(6): 1356-1367.

[3] DUTTA P, AOKI P M, KUMAR N, et al. Common sense: Participatory urban sensing using a network of hand-held air quality monitors [C]// ACM Conference on Embedded Networked Sensor Systems. 2009.

[4] GIL D S, D'OREY P M, AGUIAR A. On the challenges of mobile crowdsensing for traffic estimation [C]// ACM Conference on Embedded Networked Sensor Systems. 2017.

[5] QIN Z, FANG Z, LIU Y, et al. A Measurement Framework for Explicit and Implicit Urban Traffic Sensing [J]. ACM Transactions on Sensor Networks, 2021, 17(4): 1-27.

[6] LIU C H, PIAO C, TANG J. Energy-efficient UAV crowdsensing with multiple charging stations by deep learning [C]// IEEE INFOCOM. 2020: 199-208.

[7] BARKA E, KERRACHE C A, LAGRAA N, et al. Behavior-aware UAV-assisted crowd sensing technique for urban vehicular environments [C]// IEEE Annual Consumer Communications Networking Conference (CCNC). 2018: 1-7.

[8] TAO C, ZHU K, CHEN B, et al. UAV-assisted ground signal map construction based on 3-d spatial correlation [C]// IEEE Global Communications Conference. 2020: 1-5.

[9] ZHANG S, WU J, LU S. Collaborative mobile charging [J]. IEEE Transactions on Computers, 2015, 64(3): 654-667.

[10] LOWE R, WU Y, TAMAR A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments [C]// NIPS. 2017: 6379-6390.

[11] LIU C H, DAI Z, ZHAO Y, et al. Distributed and Energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning [J]. IEEE Transactions on Mobile Computing, 2021, 21(1): 130-146.

[12] LIU C H, CHEN Z, ZHAN Y. Energy-efficient distributed mobile crowd sensing: A deep learning approach [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1262-1276.

[13] YANG Y, RUI L, LI M Z. MING, et al. Mean field multi-agent reinforcement learning [C]// The 35th International Conference on Machine Learning. 2018: 5571-5580.

[14] WANG Z B, HU J H, LV R Z, et al. Personalized Privacy-Preserving Task Allocation for Mobile Crowdsensing [J]. IEEE Transactions on Mobile Computing, 2018, 18(6): 1330-1341.

[15] WANG L, ZHIWEN Y U, GUO B, et al. Mobile crowd sensing task optimal allocation: a mobility pattern matching perspective [M]// Frontiers of Computer Science (print), 2018: 231-244.

[16] ZHANG B, LIU C H, TANG J, et al. Learning-based energy-efficient data collection by unmanned vehicles in smart cities [J]. IEEE Transactions on Industrial Informatics, 2018, 14(4): 1666-1676.

[17] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with deep reinforcement learning [R]. Computer Science, 2013.

[18] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous methods for deep reinforcement learning [C]// Proceedings of The 33rd International Conference on Machine Learning. 2016: 1928-1937.

[19] LILLICRAP T, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning [C]// CoRR. 2016.

[20] SONG M K, WANG Z B, ZHANG Z F, et al. Analyzing User-Level Privacy Attack Against Federated Learning [J]. IEEE Journal on Selected Areas in Communications, 2020, 38(10): 2430-2444.

[21] WEI Y, ZHENG R. Multi-robot path planning for mobile sensing through deep reinforcement learning [C]// IEEE INFOCOM. 2021.

[22] DING R, YANG Z, WEI Y, et al. Multi-agent reinforcement learning for urban crowd sensing with for-hire vehicles [C]// IEEE INFOCOM. 2021.

[23] JAIN R. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems [R]. DEC Research Report, 1984.

[24] VANSTEENWEGEN P, SOUFFRIAU W, OUDHEUSDEN D V. The orienteering problem: A survey [J]. European Journal of Operational Research, 2011, 209(1): 1-10.

Cui ZHANG, born in 1998, postgraduate, is a member of China Computer Federation. Her main research interests include mobile crowdsensing and multi-agent reinforcement learning.

Funing YANG, born in 1987, Ph.D. Her main research interests include mobile computing, crowd intelligence, and data mining.

(Responsible editor: Yahui LI)