DRL-Based UAV Serving and Charging Scheduling for Autonomous Service Provisioning

Shao-Chun Huang, Li-Hsing Yen, and Yan-Wei Chen
Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

2025 International Conference on Information Networking (ICOIN) | 979-8-3315-0694-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICOIN63865.2025.10992931

***Abstract*—Unmanned aerial vehicles (UAVs) have been used to provide wireless access service to a specific area. However, the service is subject to the limited energy capacity of UAVs. One possible treatment is to arrange a serving/charging schedule for UAVs that balances the serving and charging activities to maximize the service coverage. In this study, we seek a decentralized approach where each UAV autonomously decides its action based on partial observation of the environment. Prior works either demanded complete information, which may not be feasible, or performed worse than those exploiting complete information. This paper proposes a deep reinforcement learning approach empowered by the advantage actor-critic (A2C) algorithm and long short-term memory technique (LSTM). This approach is not sensitive to the number of serving UAVs. Simulation results show that the proposed approach generally achieved higher coverage ratios compared to the counterparts.**

I. Introduction

Unmanned aerial vehicles (UAVs) have been finding applications in diverse fields ranging from surveillance and reconnaissance to target localization and area coverage. In particular, UAVs have evolved as dependable aerial base stations (BSs) for delivering network services in inaccessible areas. This ability is valuable when traditional infrastructure is overloaded or impaired due to natural disasters or military activities. However, the limited energy capacity of UAVs confines their service duration.

Overlooking the limited energy capacity of UAVs when deploying them as aerial BSs is impractical for real-world scenarios. To provide sustainable service, UAVs should return to a charging station for energy replenishment. Serving and charging activities are conflicting and complementary at the same time. On the one hand, a UAV needs adequate energy to serve for a long time. On the other hand, a UAV has less time to serve if it wastes too much time on charging or flying back and forth. It is non-trivial for a UAV to find an optimal schedule for serving and charging. The problem gets complicated when a swarm of UAVs is involved in the decision process because UAVs may have conflicting interests due to contending service rewards and limited supplies of charging facilities. It becomes more challenging when these UAVs do not communicate with each other and have limited observability, i.e., they cannot perceive complete information about the environment (serving targets and other UAVs).

Some researchers have targeted the coverage path planning (CPP) problem [1] [2], aiming at finding the most energy-efficient paths for a UAV swarm to fully cover a set of targets or a field. They typically formulated it as an optimization problem with centralized solutions. These approaches are susceptible to a single point of failure and do not adapt to evolving service demands or emerging hotspots. We thus focus on a decentralized approach where each UAV autonomously makes scheduling decisions. We model the problem as a decentralized partially observable Markov decision process (dec-POMDP) with a goal for a UAV to maximize its accumulated service reward. Our work was inspired by [3], which used a deep recurrent Q-network (DRQN) algorithm to maximize coverage in a target area. Unfortunately, their results were inferior to a prior game-theoretic work [4] regarding service coverage ratio. Therefore, we propose replacing DRQN with an Advantage Actor-Critic (A2C) algorithm for performance improvement. Like [3], we use the long short-term memory (LSTM) [5] technique in our neural network to effectively handle and understand sequential data, which is valuable in our scenario as our UAVs have limited observations. Our approach incorporates two advanced techniques. The first technique involves storing recurrent states, while the second involves reserving a portion of the training sequences for a burn-in period [5].

The contributions of this paper follow:

- We propose a decentralized method based on deep reinforcement learning (DRL) for individual UAV with partial observation to autonomously determine its charging and serving schedule.
- The method enhances the resilience of UAV service as each UAV's behavior and performance is resistant to the variation of the system (e.g., the size of the UAV swarm). This feature provides us the flexibility to adjust the number of on-duty UAVs dynamically to adapt to time-varying service demands.
- Simulation results indicate that our method yields a higher coverage ratio compared to prior work [4, 3] under different numbers of on-duty UAVs.

The rest of the paper is organized as follows. Sec. II reviews related work. Sec. III presents our system model. Then, Sec. IV formulates the UAV scheduling problem. In Sec. V, we introduce our algorithm and some design details. Sec. VI demonstrates the simulation results compared to other methods. We conclude this paper in Sec. VII.

II. Related Work

The issue of UAV deployment has recently been the subject of widespread scholarly attention. A range of solutions has

been suggested for diverse scenarios and settings. This section provides an overview of these studies.

A. Service Coverage Optimization

Several studies assumed that ground users are all under the signal coverage of UAVs [6, 7], overlooking the fact that complete coverage may not always be achievable in real-world scenarios. Some other studies attempted to maximize the ratio of the target field that UAVs cover [8] [9].

Many studies [1] [10] have explored UAV missions that provide complete services or surveillance coverage over target areas. Some studies [11, 6] considered UAV energy capacity in providing seamless coverage over a target area. However, these methods assumed uniform network demands across the target area. By contrast, we consider maximizing service coverage over a target area with non-uniform service demands and a fixed number of UAVs (each with a limited energy supply). Our approach can provide seamless coverage when sufficiently many UAVs are involved.

B. Energy Efficient UAV Control and Charging Strategies

Some studies (e.g., [12]) considered UAV energy constraints in delivering network services to a specific area or set of users. However, energy-efficient designs may not guarantee sustainable network service if the designs do not consider UAV energy replenishment by battery replacement or recharging. The concept of recharging scheduling has been demonstrated in numerous studies [3] [13]. Liu et al. [14] coordinated UAVs to execute recharging and service delivery to maximize data collected by the UAVs. Han et al. [15] planned a UAV recharging schedule to ensure extended area monitoring. Sun et al. [16] proposed a charging and service strategy to provide agricultural plant protection.

C. Deep Reinforcement Learning

Several researchers have designed centralized algorithms to direct UAV missions [16, 8]. Many researchers have shifted towards a decentralized approach for a more practical and robust design. One direction of the distributed method is game-theoretic designs [10, 17, 4]. However, these designs typically require complete information for the game process, which may not always be feasible in real-world situations.

In response, some researchers have proposed alternative methods grounded in DRL [11, 12, 6, 18]. In particular, Chen and Yen [3] proposed a DRL-based approach for the game-theoretic design in [4] to eliminate the requirement for complete information. Unfortunately, their solution exhibited inferior service coverage radios compared to [4]. Our work is a follow-up design of [3] that takes the same system model and problem formulation but uses an alternative approach.

III. System Model

We modified the system model in [3]. We assume a set of U UAVs $\mathcal{U} = \{1, 2, \ldots, u, \ldots, U\}$. We divide the service time into consecutive time slots $\mathcal{T} = \{1, 2, \ldots, t, \ldots\}$. A UAV can choose one of four possible actions in each time slot. *Serving* to deliver service to the current area. *Recharging* is to recharge the battery at a charging station. *Flying* is to fly to a different area for serving or charging. *Resting* is to standby at the current location. These actions account for associated micro-actions such as landing or taking-off needed for possible transitions between actions.

The target area is equally divided into subareas $\mathcal{A}$. Each subarea $a \in \mathcal{A}$ requests d_a UAVs to fulfill its service demand. Charging stations are set up in some subareas $\mathcal{M} \subseteq \mathcal{A}$. We use $C(a)$ to indicate where a charging station is located in subarea $a \in \mathcal{A}$. There are l_m charging slots in each charging station m, which permits at most l_m UAVs to recharge simultaneously.

We denote the location of UAV u at time slot t as $\mathcal{L}_u^t \in \mathcal{A}$. We denote the sets of subareas that UAV u can observe and serve at time t as $\mathcal{O}_u^t$ and $\mathcal{S}_u^t$, respectively. When UAV u is in subarea a, the set of subareas that UAV u can relocate in the next time slot is denoted by $\mathcal{R}_u(a)$.

The energy capacity and the remaining energy of each UAV u at time t are $b_u^{\max}$ and b_u^t joules (J), respectively, with $0 \leq b_u^t \leq b_u^{\max}$. The energy charging rate of UAV u is k_u J per time slot. The energy consumption of a UAV depends on its actions. We denote the energy consumption of serving and flying for a time slot as e_u^{ser} and e_u^{fly} J, respectively.

IV. Problem Formulation

We formulate the coverage maximization problem and present the decentralized partially observable Markov decision process (dec-POMDP).

A. Maximizing Accumulated Service Satisfaction Level

We use c_u^t, x_u^t, and f_u^t to indicate whether UAV u is charging, serving, and flying, respectively, in time slot t. We use $\mathcal{L}_{u,a}^t \in \{0, 1\}$ to denote whether UAV u is in subarea a at time t. The mission and charging scheduling problem is to determine the values of c_u^t, x_u^t, f_u^t, and $\mathcal{L}_{u,a}^t$, for each UAV u in each time slot t to maximize the total accumulated service satisfaction level (SSL) across the entire target area $\mathcal{A}$ over time interval $\mathcal{T}$:

$$\max_{c_u^t, x_u^t, f_u^t, \mathcal{L}_{u,a}^t} \sum_{t \in \mathcal{T}} \sum_{a \in \mathcal{A}} z_a^t, \tag{1}$$

where z_a^t is the SSL in subarea $a \in \mathcal{A}$ at time t. The value of z_a^t is d_a if at least d_a UAVs serve a at time t. It is zero otherwise. This definition gives a subarea an SSL proportional to its service demand if the demand is satisfied.

The objective is subject to the following constraints:

$$\sum_{a \in \mathcal{A}} \mathcal{L}_{u,a}^t = 1 \quad \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \tag{2}$$

$$\sum_{u \in \mathcal{U}} (c_u^t \times \mathcal{L}_{u,a}^t) \leq l_m, \quad \forall a \in \mathcal{M}, \forall t \in \mathcal{T}, \tag{3}$$

$$x_u^t \times e_u^{\text{ser}} \leq b_u^{t-1}, \quad \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \tag{4}$$

$$f_u^t \times e_u^{\text{fly}} \leq b_u^{t-1}, \quad \forall u \in \mathcal{U}, \forall t \in \mathcal{T}, \tag{5}$$

$$c_u^t + x_u^t + f_u^t \leq 1, \quad \forall u \in \mathcal{U}, \forall t \in \mathcal{T},$$
$$c_u^t \leq C(\mathcal{L}_u^t), \quad \forall u \in \mathcal{U}, \forall t \in \mathcal{T},$$
$$\mathcal{L}_u^{t+1} \in \mathcal{R}_u(\mathcal{L}_u^t), \quad \forall u \in \mathcal{U}, \forall t \in \mathcal{T},$$

Eq. (2) asserts that a UAV can only be located in a subarea. Eq. (3) ensures that the number of UAVs concur charging at one station cannot exceed the number of ava charging slots. The residual energy of UAV u at the e time t can be calculated by

$$b_u^t = b_u^{t-1} + c_u^t \times k_u - x_u^t \times e_u^{ser} - f_u^t \times e_u^{fly},$$

where b_u^{t-1} represents the remaining battery level of U at time $t-1$, and the other terms express the energy reg from charging, consumed while serving, and expende ing movement, respectively. Eq. (4) and (5) impose e constraints on the UAVs, prohibiting them from runnir of energy by serving and flying, respectively. Since a UAV can perform only one action at a time, we have the constraint $c_u^t + x_u^t + f_u^t \leq 1$ as shown in (6). When c_u^t, x_u^t, and f_u^t are all zeros, the UAV chooses to rest. Eq. (7) specifies that UAVs can only recharge at designated charging stations. Finally, Eq. (8) enforces that a UAV's velocity cannot exceed its defined limit.

B. Dec-POMDP

A UAV's decision-making can be modeled as a Markov decision process (MDP), where the UAV as an agent chooses an action a based on the current state s of the system. The action causes a state transitions to a new state s' with probabilities $T(s, a, s')$ and a corresponding reward $R(s, a, s')$ for the agent. Because an agent's reward depends on other agents' actions, this situation forms a stochastic game, and the decision-making of all UAVs becomes a multi-agent MDP (MMDP). We assume UAVs make decisions independently, turning the problem into a decentralized MMDP. Solving decentralized MMDP is challenging, as agents' policies may not converge to a stable yet optimal result. For stability, this game has been proven an exact potential game (EPG) [4]. To reduce possible state space, we assume homogeneous UAVs, attempt training an MDP, and let each UAV use the same MDP to make its own decision. The optimal MDP provides an action for a general agent in each state to maximize the agent's expected reward over a potentially infinite horizon.

Since UAVs only have partial observation, we refer to decentralized, partially observable MDP (dec-POMDP). In a dec-POMDP, every agent i holds an individual policy π_i that maps the history of its observations and actions $H_i^t = o_i^1, a_i^1, o_i^2, a_i^2, \ldots, o_i^{t-1}, a_i^{t-1}, o_i^t$ into respective actions a_i^t. Here, o_i^t and a_i^t signify the observation received and the action taken by agent i, respectively, at time t. The policy space $\pi = \Pi_i \pi_i$ collects all combinations of individual policies. The goal is to identify the optimal joint policy $\pi^* \in \pi$ that maximizes the cumulative reward as indicated in (1).

We detail the components of our dec-POMDP model as follows. The state space at time slot t consists of $\{\mathcal{L}_u^t\}_{u \in \mathcal{U}}$, $\{b_u^t\}_{u \in \mathcal{U}}$, $\{d'_a\}_{a \in \mathcal{A}}$, and $\{l'_m\}_{m \in \mathcal{M}}$, where d'_a is the number of UAVs still needed to fulfill the service demand in subarea a and l'_m is the number of available charging slots in charging station m. The information that each UAV u can observe at time slot t includes b_u^t, $\{d'_a\}_{a \in \mathcal{O}_u^t}$, and $\{l'_m\}_{m \in \mathcal{O}_u^t}$, where $\mathcal{O}_u^t$ is the set of subareas observable by u at time slot t. The action space of each UAV is $\{Serving, Moving, Charging, Resting\}$.

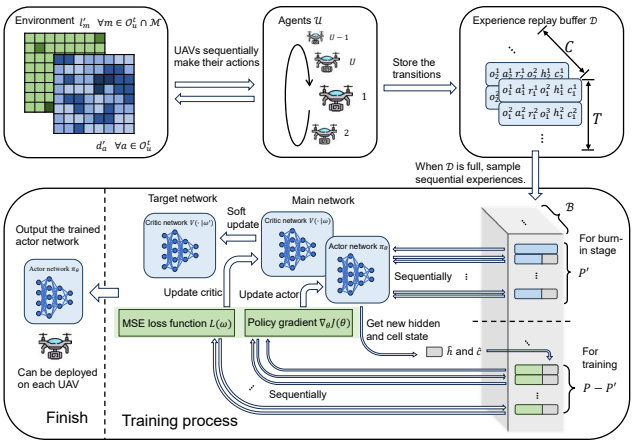


Figure 1: Our training framework.

V. Proposed Approaches

We take the centralized training distributed execution (CTDE) framework, where a server oversees the learning processes of all agents, and each agent independently executes its policy within the environment. In the training phase, the server compiles all agents' experiences within a shared replay buffer. These experiences are then used to update policies and value functions across all agents. We decouple target values from the updating parameters, which effectively addresses the training instability caused by the shifting target problem. After the training, each agent obtains the updated policy from the server. The fact that the trained policy universally applies to all agents makes UAV deployment flexible, as it facilitates the addition of extra UAVs into the target area. Fig. 1 depicts our training framework.

A. Model Architecture

We use two policies for UAVs: one for high energy levels and the other for low energy levels. These policies are implemented as two neural networks (NNs), which take $\{d'_a\}_{a \in \mathcal{O}_u^t}$ and $\{l'_m\}_{m \in \mathcal{O}_u^t}$, respectively, as inputs. Each NN uses two convolutional layers (CLs) to interpret the observations, followed by an LSTM layer to facilitate decision-making using historical data. Each NN concludes with a fully connected (FC) layer at the end to produce the output.

We set up $\mathcal{E}$ episodes for the training procedure, each comprising T discrete time slots. At the beginning of each episode, we initialize relevant settings such as subareas' service demands d_a and charging station locations. The UAVs are initially dispersed over the target field to provide training diversity as each UAV u's location determines its initial

observation o_u^{init}. UAVs then take turns to act, one at each time slot. UAV u's selection of its action a_u^t in time slot t is based on its observation o_u^t while accounting for some random noise $\mathcal{N}$ to promote exploration (i.e., not sticking to deterministic actions). The action causes a state transition, and u receives the corresponding reward r_u^t by the following rule.

- If a_u^t is flying, u receives a high penalty if it moves out of the target field and a small reward otherwise.
- If a_u^t is serving and $b_u^t > e_u^{\text{ser}}$, r_u^t sums up a reward $r_u^t(a)$ from each subarea $a \in \mathcal{S}_u^t$. The value of $r_u^t(a)$ is d_a if $z_a^t \geq d_a$ and u is among the first d_a UAVs that serve a. The value is zero otherwise.
- r_u^t is a penalty if u chooses flying or serving but $b_u^t \leq e_u^{\text{ser}}$ or $b_u^t \leq e_u^{\text{fly}}$.
- If a_u^t is charging, $b_u^t < b_u^{\max}$, and $l'_a \geq 1$, u receives a reward according to b_u^t. It receives a penalty if b_u^t equals $b_u^{\max}$ or $l'_a = 0$.
- u receives no reward in all other cases.

The current observation, action, reward, the observation of the following state o_u^{t+1}, and LSTM's hidden and cell states (h_u^t and c_u^t), are kept in the shared replay buffer $\mathcal{D}$ as a transition $(o_u^t, a_u^t, r_u^t, o_u^{t+1}, h_u^t, c_u^t)$. All T transitions in an episode are collected as a *full trajectory*.

The replay buffer can hold C full trajectories. When the replay buffer gets full (i.e., after C episodes), we use it to perform experience replay to update our policy network.

B. Advantage Actor-Critic Algorithm (A2C)

In the advantage actor-critic (A2C) reinforcement learning algorithm, the actor determines the action to take in each state, while the critic estimates the advantage of the chosen action. The ultimate goal is to find the optimal policy that maximizes the agent's total accumulated reward

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t), \tag{10}$$

where $R(s_t, a_t)$ is the reward of taking action a_t in state s_t and $\gamma \in [0, 1]$ is the discount factor.

For an agent's policy $\pi(a|s)$ that takes action a in state s at time t, the expected return is

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \,\middle|\, s_t = s, a_t = a \right] \tag{11}$$

while the average expected return starting from state s is

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \,\middle|\, s_t = s, \pi \right]. \tag{12}$$

We use the advantage function $A_\pi(s, a)$ to capture the effect of taking action a in comparison to the average return in the current state s:

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s). \tag{13}$$

The emphasis is to approximate solely the state value function $V_\pi(s)$ rather than both $Q_\pi(s, a)$ and $V_\pi(s)$. Because $Q_\pi(s, a)$ can be reformulated as $Q_\pi(s, a) = \mathbb{E}\left[R(s_t, a_t) + \gamma V_\pi(s_{t+1}) \mid s_t = s, a_t = a, \pi\right]$, we have the following equation:

$$A_\pi(s, a) = \mathbb{E}_\pi \left[R(s_t, a_t) + \gamma V_\pi(s_{t+1}) \mid s_t = s, a_t = a, \pi\right] - V_\pi(s). \tag{14}$$

The training aims to enhance the policy $\pi_\theta(a|s)$, with θ representing the policy network parameters. We apply the advantage function to update the policy parameters:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J(\theta), \tag{15}$$

where

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A_\pi(s_t, a_t) \right] \tag{16}$$

and α_θ is the learning rate of the policy network. We also use the advantage function to be our critic network model. Therefore, the loss of a critic network parameters ω is

$$L(\omega) = \mathbb{E}_\pi \left[R(s_t, a_t) + \gamma V_\pi(s_{t+1} \mid \omega) - V_\pi(s_t \mid \omega)\right]^2. \tag{17}$$

Consequently, we update the critic parameters ω by

$$\omega \leftarrow \omega + \alpha_\omega \nabla_\omega L(\omega), \tag{18}$$

where α_ω represents the critic network's learning rate.

C. The Training Algorithm

Our prior work [3] performs experience replay using complete trajectories, which can cause some potential issues as pinpointed in [19]. Therefore, we randomly select trajectory segments for training. Specifically, we randomly sample a batch of $\mathcal{B}$ trajectory segments from $\mathcal{D}$. Each segment is a sequence of P consecutive transitions. The first $P' < P$ transitions in the segment are reserved for a burn-in stage, which is to unroll the network and generate a start state. The rest are then used to train our policy network.

For a random segment ending at time slot t', the transitions starting from time slot $t' - P$ to $t' - P + P'$ are replayed to derive the hidden state $h_u^{t'-P+P'}$ and cell state $c_u^{t'-P+P'}$ of the LSTM in the burn-in state. The newly updated states $\hat{h}_u^{t'-P+P'}$ and $\hat{c}_u^{t'-P+P'}$ are then used to update the actor and critic networks with the rest of the segment.

Critic updates involve minimizing the mean squared error (MSE) loss, expressed as $L(\omega) = \frac{1}{|\mathcal{B}|} \sum_{\mathcal{B}} (r^k + \gamma V_\pi(o^{k+1}|\omega') - V_\pi(o^k|\omega))^2$, where k iterates from the first to the last transition in the rest of the segment. Meanwhile, actor updates rely on sampled policy gradients: $\nabla_\theta J(\theta) = \frac{1}{|\mathcal{B}|} \sum_{\mathcal{B}} \nabla_\theta \log \pi_\theta(a^k|o^k) A_\pi(o^k, a^k)$. Following the sequential updates, target network parameters are updated using the soft update equation $\omega' = \omega + (1-\tau)\omega'$, where τ is the soft update parameter.

VI. Simulation Results

We evaluate the performance of our algorithm using simulations. Two main measurements were service coverage ratio and average residual energy. The performance of our approach was compared to those of two counterparts: one based on the EPG [4] and the other based on DRQN [3]. For a comparison, we augmented DRQN with state storing and burn-in techniques.

A. Simulation Setting

We considered a grid of 10×10 square subareas, with each subarea measuring 25 meters on each side. Each subarea demanded the service from two (uniformly) or zero to seven (non-uniformly) UAVs by default. We deployed five charging stations, each with 3 to 9 charging slots. We deployed a swarm of 4 to 40 UAVs. Each UAV had a service radius of 62.5 meters and an observation range of 112.5 meters. Each UAV could move to one of the 3×3 neighboring subareas for its following location. The battery capacity of the UAV was set to 200 kilojoules (kJ). It took a UAV of 1 kJ for service or flying to a subarea within a time slot. The charging rate was 0.5 to 2 kJ per time slot.

Our model training encompassed 100 episodes, with each episode comprising 1000 time slots. The replay buffer was capable of storing 400 episodes. The batch size was 8, coupled with sequence length $P = 20$ and burn-in sequence length $P' = 5$. We updated the actor and the critic with learning rates of 0.001 and 0.002, respectively. The soft update parameter was $\tau = 0.005$.

B. Uniform Service Demand Distribution

We first set the service demand to $d_a = 2$ for all $a \in \mathcal{A}$. Fig. 2a to Fig. 2c depict how the service coverage ratio improved with the number of UAVs and charging rate k_u, where the ratio is defined as the accumulated SSL to the total demands over the entire duration of the simulation. When $k_u = 0.5$, the EPG approach outperformed DRQN and A2C. However, its superiority disappeared when 12 or more UAVs were involved with $k_u \geq 1.0$.

Fig. 2d provides insights into the average residual energy ratio, i.e., the average ratio of b_u^t to $b_u^{\max}$ throughout the experiment. The EPG approach differed significantly from the others. With a small number of UAVs, UAVs in EPG tended to engage in service provision or movement, leading to a higher coverage ratio but lower residual energy compared to the others. When UAVs using EPG did not observe service demands, which occurred when a large number of UAVs were involved, they tended to perform charging to gain a charging reward. By contrast, both DRQN and A2C exhibited a consistent trend in energy preservation, regardless of the number of UAVs in the field. UAVs in these approaches continuously traversed the target field, searching for demands independently of the presence of other UAVs or the charging rate. This persistent movement consumed individual UAV energy but increased the overall coverage ratio.

C. Non-Uniform Service Demand Distribution

We also conducted experiments with a clustered service demand distribution. Specifically, two demand clusters were on the field, each centered around regions with the highest demand, gradually decreasing with distance from the center.

Fig. 3a to Fig. 3c illustrate the average coverage ratio concerning the charging rate and the number of UAVs. Compared to the results shown in Fig. 2, the performance of EPG reduced significantly while those of DRQN and A2C did not. In fact, the latter two algorithms even had improved performance when few UAVs were involved. This result can be justified by UAV residual energy shown in Fig. 3d, where DRQN and A2C exhibited a consistent energy level, similar to previous observations. By contrast, UAVs in EPG took too much time to charge while there were still service demands to explore, resulting in a high residual energy level but a low coverage ratio.

VII. Conclusions

We have proposed a decentralized method based on A2C and LSTM for UAVs with partial observation to determine its charging and serving schedule autonomously. Our experimental results have demonstrated that the proposed approach outperforms the DRQN-based approach regarding coverage ratio. The proposed approach outperformed the EPG-based approach in case of a high charging rate or non-uniform service demand distribution.

It is worth noting that we have not explicitly addressed the altitude factor for UAVs in our setting. Considering the altitude factor would add realism to the challenges faced in real-world applications, and it stands as our future work.

Acknowledgment

This work has been supported by the National Science and Technology Council of Taiwan under grant numbers NSTC 113-2221-E-A49-180 and NSTC 113-2218-E-A49-027.

References

[1] S. D. Apostolidis et al., "Cooperative multi-UAV coverage mission planning platform for remote sensing applications," *Autonomous Robots*, vol. 46, pp. 373–400, Feb. 2022.

[2] M. A. Luna et al., "Fast multi-UAV path planning for optimal area coverage in aerial sensing applications," *Sensors*, vol. 22, p. 2297, Mar. 2022.

[3] H.-C. Chen and L.-H. Yen, "DRL-based distributed joint serving and charging scheduling for UAV swarm," in *Proc. ICOIN*, Ho Chi Minh City, Vietnam, Jan. 2024.

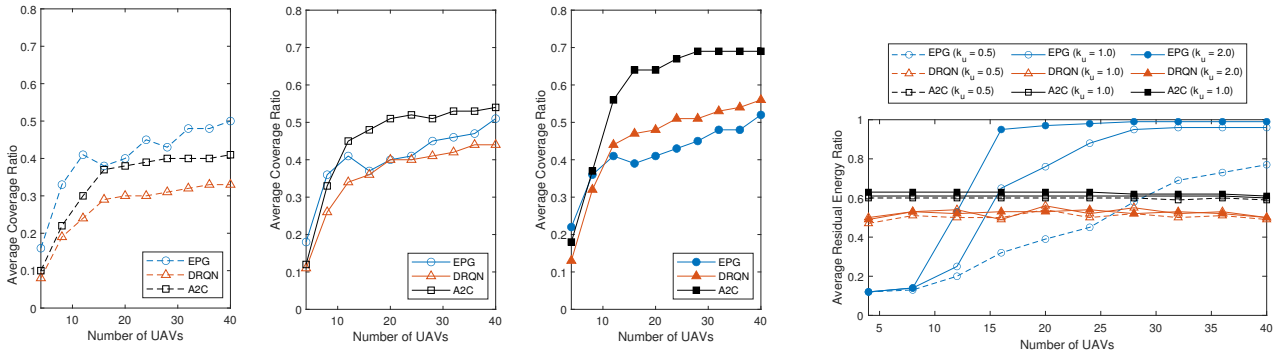
[4] C.-I. Li et al., "Distributed mission and charging scheduling for UAV swarm to maximize service coverage," in *IEEE VTC2021-Fall*, Sep. 2021.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] Y. Zhang et al., "Hierarchical deep reinforcement learning for backscattering data collection with multiple UAVs," *IEEE Internet Things J.*, vol. 8, pp. 3786–3800, Mar. 2021.

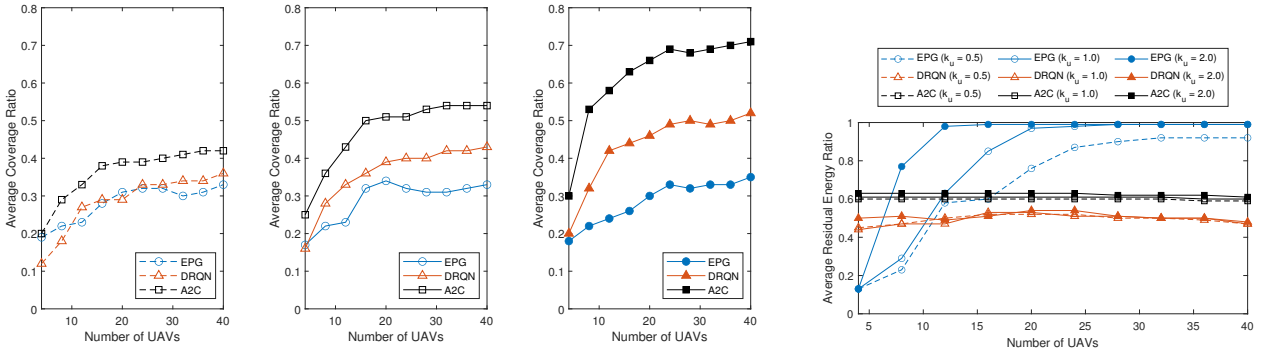
[7] W. Lee and T. Kim, "Multi-agent reinforcement learning in controlling offloading ratio and trajectory for multi-UAV mobile edge computing," *IEEE Internet Things J.*, vol. 11, pp. 3417–3429, Jul. 2023.

[8] S. Shakoor et al., "Joint optimization of UAV 3-D placement and path-loss factor for energy-efficient maximal



(a) $k_u = 0.5$ (b) $k_u = 1.0$ (c) $k_u = 2.0$ (d) All k_u's

Figure 2: (a) to (c): Average coverage ratio and (d): average residual energy with increasing number of UAVs. ($d_a = 2$)



(a) $k_u = 0.5$ (b) $k_u = 1.0$ (c) $k_u = 2.0$ (d) All k_u's

Figure 3: (a) to (c): Average coverage ratio and (d): average residual energy with increasing number of UAVs. (d_a: clustered distributed)

coverage," *IEEE Internet Things J.*, vol. 8, pp. 9776–9786, Jun. 2021.

[9] Y. Kadry et al., "Meta-heuristic algorithms for solving the 3D volume coverage problem using multi-UAVs," in *Novel Intelligent and Leading Emerging Sciences Conference*, 2022, pp. 194–199.

[10] M. G. S. De Campos et al., "Towards a blockchain-based multi-UAV surveillance system," *Frontiers in Robotics and AI*, vol. 8, no. 557692, Jun. 2021.

[11] M. Theile et al., "UAV coverage path planning under varying power constraints using deep reinforcement learning," in *Proc. IEEE/RSJ IROS*, Oct. 2020, pp. 1444–1449.

[12] C. H. Liu et al., "Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 19, pp. 1274–1285, Jun. 2020.

[13] Z. Song et al., "Methods to assign UAVs for k-coverage and recharging in IoT networks," *IEEE Trans. Mobile Comput.*, vol. 23, pp. 2504–2519, Mar. 2023.

[14] C. H. Liu et al., "Energy-efficient UAV crowdsensing with multiple charging stations by deep learning," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 199–208.

[15] Z. Han et al., "Scheduling rechargeable UAVs for long time barrier coverage," in *Proc. IEEE ICPADS*, Dec. 2020, pp. 282–289.

[16] F. Sun et al., "Task scheduling system for UAV operations in agricultural plant protection environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, May 2020.

[17] N. Gao et al., "Coverage control for UAV swarm communication networks: A distributed learning approach," *IEEE Internet Things J.*, vol. 9, pp. 19 854–19 867, Oct. 2022.

[18] Z. Ye et al., "Multi-UAV navigation for partially observable communication coverage by graph reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 22, pp. 4056–4069, Jul. 2022.

[19] S. Kapturowski et al., "Recurrent experience replay in distributed reinforcement learning," in *Proc. ICLR*, Dec. 2019.