DRL-Based Distributed Joint Serving and Charging Scheduling for UAV Swarm

Hsiao-Chi Chen and Li-Hsing Yen
Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.
Email: {magicxcr7.cs08g,lhyen}@nycu.edu.tw

***Abstract*—Unmanned aerial vehicles (UAVs) have been applied to a wide range of applications. When planning a mission for battery-powered UAVs, energy replenishment is a factor that cannot be ignored. This paper considers a decentralized scheduling scheme for a swarm of UAVs for serving and charging activities, which is challenging because of the trade-off between service requirement and energy consumption as well as limited supply of charging facilities. We propose two decentralized schemes based on deep reinforcement learning (DRL) with partial observation that allow the UAV swarm to autonomously learn where to rest, provide service, or recharge. Although the learning model is for a single UAV, it applies to each UAV in the swarm. We conducted simulations for performance measurements. The results show that the proposed approaches are feasible for distributed serving and charging scheduling with multiple UAVs.**

I. Introduction

Unmanned aerial vehicles (UAVs) have been applied to a wide range of applications. Different UAV applications and services demand different types of UAV deployments. For example, we may deployment a UAV to make a tour to visit a set of fixed spots for data collection from sensory or Internet-of-Thing (IoT) devices. To provide access and relay services in some area, we may deploy a UAV to hover and stay in a single, selected spot. Yet another UAV use case is to patrol in a disaster area to support rescue teams and survivors. In that application, a UAV may need to autonomously seek serving spots and figure out its serving priority.

Regardless of the deployment type, battery-powered UAV needs to fly back to a charging station for energy replenishment when the UAV is about to run out of its battery power. Energy replenishment can be done by battery exchange in-air charging, or ground charging. Charging a UAV generally takes considerable time and may cause service outage to the application. However, a low-battery UAV may crash in service or on the way back to the charging station. Therefore, it is crucial to appropriately arrange serving and charging time of UAV to maximize its quality of service.

A way to enhance the quality of UAV service is to deploy a swarm of UAVs. In that case, coordinating UAVs for their serving and charging activities becomes challenging because of limited supply of charging facilities. Several studies have been on the charging and serving schedule of UAVs [1], [2], [3] for different types of UAV deployments. Most existing solutions take centralized approaches to ease the coordination among UAVs, which seems a reasonable design choice and practical solution. However, centralized approaches are not always feasible and are subject to a single point of failure. Therefore, in this work we seek a decentralized approach where each UAV can autonomously determine its action to adapt to other UAV's actions and the dynamic environment.

This study particularly considers UAVs as data collection points for IoT devices. IoT devices generate service coverage demands which should be fulfilled by one or more UAVs. To allow UAVs to recharge and continue their coverage service, several round charging stations each with multiple charging slots are set up. For this setting, Li et al. [4] turned the UAV swarm scheduling problem into an exact potential game (EPG), where each UAV decides its own trajectory and service target. The limitation of this work is that each UAV needs the knowledge of other UAV's activities when making its decision.

In this paper, we propose two decentralized schemes based on deep reinforcement learning (DRL) with partial observations. These schemes allow each UAV in a UAV swarm autonomously learns where to rest, provide service, or recharge without the knowledge of other UAV's activities. The learning model is for a single UAV but applies to each UAV, thus achieving scalability. We conducted simulations which considered the impact of the number of UAVs, clustered distribution of service demands, and charging capacities. The results show that the proposed approaches can handle distributed serving and charging scheduling with adequate charging capacity.

The remainder of this paper is organized as follows. We review related work in Sec. II. Sec. III presents the system model and formulates the problem. The proposed approaches, DQN and DRQN, are presented in Sec. IV. The next section reports simulation results and that last section concludes this paper.

II. Related Work

UAV placement is to decide the serving locations for UAVs. Many UAV placement approaches considered location-varying energy consumption to maximize service coverage or transmission rates Hu et al. [5] proposed a sensing-and-send protocol and subchannel allocation mechanism to maximize the probability of successful sensory data transmission to UAVs. Pham et al. [6] proposed a multi-agent reinforcement learning to maximize the sensing coverage. The approach proposed by Hu et al. [7] divides users into clusters and finds UAV trajectories that maximize coverage of the clusters.

Other studies further considered power consumption on flying for better UAV flight directions [8] or flight trajectories

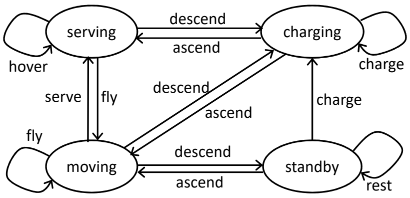


Fig. 1: Operation modes of a UAV

[9]. Ding et al. [10] developed a strategy to achieve continuous coverage and found the optimal speed configuration for path planing to minimize power consumption of flight. Zhang et al. [11] took the signal-to-noise ratio received by the user as a coverage constraint and found the 3D trajectory to maximize the link capacity while satisfying the coverage constraint. Game theory can be well applied to the selection strategy among UAV swarm. Ruan et al. [12], [13] proposed a distributed approach based on game theory to select a strategy for UAV swarm that maximizes coverage area and minimizes flight power consumption.

All above-mentioned studies ignore the need for charging. Ghazzai et al. [14] and Khosiawan et al. [15] found optimal schedule for charging, task, flight, and waiting events to meet both energy constraint and the task requirement. Shakhatreh et al. [1] proposed a heuristic to find the minimum number of UAVs for continuous coverage considering both power consumption on flight and charging. Qi et al. [16] proposed a DRL-based approach to maximize communication data volume and minimize UAVs energy consumption while achieving fair and persistent service coverage on users.

All the above approaches are centralized. For decentralized approaches, many studies applied game theory to solve the problem [17], [4]. Trotta et al. [17] considered using rechargeable UAVs to autonomously cover a target area and replenish energy when needed. They proposed a distributed game-theoretic scheduling strategy to maximize the stationary coverage and to guarantee the continuity of the service. Li et al. [4] extended the work in [17] by considering multiple subareas, each having a specific coverage requirement, and different numbers of charging stations. Their experimental results showed that the game-theoretic approaches can strike a balance between charging and coverage, and adapt well to different environmental settings.

This paper follows the system model of Li et al. [4] and jointly addresses the serving and charging scheduling problem for a UAV swarm through DRL. The proposed approach differs from Li et al.'s work [4] in that a UAV in our approach does not need complete information about all other UAVs and all working areas to decide its action. Although reinforcement learning or DRL has been used for various missions in UAV, our work is unique in the problem setting.

III. System Model and Problem Formulation

A. System Model

We assume a swarm of n rechargeable UAVs $P = \{1, 2, \ldots, n\}$ to provide access service over a target area. The operation time consists of t equal time slots $T = \{1, 2, \ldots, t\}$. In each time slot, a UAV can be in one of four operation modes: serving, charging, flying, and standby. Initially, all UAVs are standby. A UAV can then ascend, which causes the UAV to enter flying mode, and fly to a spot to serve ground terminals (thus entering serving mode). A serving UAV may decide to cease the service and then fly (entering flying mode) to another spot for serving or charging. Because the altitude of charing station is much lower the hovering altitude of UAV, a UAV needs to descend before it can actually access the charging facility (entering charging mode). For the same reason, a UAV after charging needs to ascend first. It can then stays in the same area or flies to another spot to provide access service. Fig. 1 shows the operation transition diagram of a UAV.

The target area is composed of a set of subareas A. The size of a subarea is sufficiently small to be served by a UAV while a UAV may serve several subareas simultaneously. The set of subareas served by UAV p in time slot $t \in T$ is denoted by $Cov_p^t \subseteq A$, which depends on the location of p in time slot t. We assume that each UAV has a limited observation range and use Sen_p^t to denote the set of subareas that UAV i can observe in time slot t. Sen_p^t is generally a superset of Cov_p^t.

We assume that the demand for access service varies in intensity geographically. To capture the degree of intensity, we use d_a to denote the number of UAVs that are needed to fulfill the service demand in a subarea $a \in A$.

We assume one or more charging stations dispersed over A. At most one charging station resides in each subarea and we use $M \subseteq A$ to represent the set of subareas where a charging station resides. We use $C(a) \in \{0, 1\}$ to indicate whether a charging station resides in subarea $a \in A$. Each charging station $m \in M$ has s_m charging slots, where one charging slot allows for one UAV to charge. We assume that each UAV $p \in P$ has a battery capacity b_p^{Max} J. and charging rate k_p J. per time slot. The residual battery level of UAV p at the end of time slot t is denoted by b_p^t, where $0 \leq b_p^t \leq b_p^{\text{Max}}$. The energy consumption rate of a UAV when providing access service is e_p^{ser} J. per time slot. The energy consumption rate of a UAV hovering in a subarea or flying to another subarea is e_p^{fly} J. per time slot.

B. Problem Formulation

This subsection formulates the joint serving and charging scheduling problem for UAV swarm. For each UAV $p \in P$, we use $x_p^t \in \{0, 1\}$, $y_p^t \in \{0, 1\}$, and $f_p^t \in \{0, 1\}$ to indicate whether p is charging, serving, and flying (or hovering), respectively, in every time slot $t \in T$. A UAV p is idle in time slot t if $x_p^t = y_p^t = f_p^t = 0$. As p is in exactly one of these four possibles operation modes at any time, we have $x_p^t + y_p^t + f_p^t \leq 1$ for all $p \in P$ and $t \in T$. With these indicator variables, the residual energy of UAV p at the end of time slot t can be estimated as follows:

$$b_p^t = b_p^{t-1} + x_p^t \times k_p - y_p^t \times e_p^{\text{ser}} - f_p^t \times e_p^{\text{hov}}, \forall p \in P, \forall t \in T, \quad (1)$$

where b_p^{t-1} is the residual energy of p at the end of time slot $t-1$. The term $x_p^t \times k_p$ represents the energy gain if p charges in time slot t. The last two terms represent the energy consumption of p if it serves and flies, respectively, in time slot t.

Our goal is to maximize the accumulated service coverage intensity. The service coverage intensity of a subarea a is d_a if d_a or more UAVs cover a and zero otherwise. We use z_a^t to denote the coverage intensity of subarea a in time slot t. Formally,

$$z_a^t = \begin{cases} d_a, & \text{if } \sum_{p \in U^t(a)} y_p^t \geq d_a, \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where $U^t(a) = \{p \mid p \in P : a \in Cov_p^t\}$ is the set of the UAVs that are able to cover subarea a in time slot t.

The objective function sums up the service coverage intensities of the whole area over the entire operation time T:

$$\max_{x_p^t, y_p^t, f_p^t, Loc_{pa}^t} \sum_{t \in T} \sum_{a \in A} z_a^t, \quad (3)$$

where Loc_{pa}^t indicates whether UAV p resides in subarea $a \in A$ in time slot t. The objective is subject to the following constraints:

$$\sum_{a \in A} Loc_{pa}^t = 1, \forall p \in P, \forall t \in T, \quad (4)$$

$$\sum_{p \in P} \left(x_p^t \times Loc_{pm}^t\right) \leq s_m, \forall m \in M, \forall t \in T, \quad (5)$$

$$(y_p^t \times e_p^{\text{ser}}) \leq b_p^t, \forall p \in P, \forall t \in T, \quad (6)$$

$$f_p^t \times e_p^{\text{hov}} \leq b_p^t, \forall p \in P, \forall t \in T, \quad (7)$$

$$x_p^t + y_p^t + f_p^t \leq 1, \forall p \in P, \forall t \in T, \quad (8)$$

$$x_p^t \leq C(Loc(p,t)), \forall p \in p, \forall t \in T, \quad (9)$$

$$x_p^t + y_p^t + f_p^t \geq 1 - C(Loc(p,t)), \forall p \in P, \forall t \in T, \quad (10)$$

$$Loc(p, t+1) \in R_p(Loc(p,t)), \forall p \in P, \forall t \in T, \quad (11)$$

where $Loc(p,t) = a$ if and only if $Loc_{pa}^t = 1$, and $R_p(a)$ denotes the set of subareas that UAV p can reach in the next time slot when p is currently in subarea a.

Eq. (4) asserts that each UAV stays in exactly one subarea in any time. Eq. (5) states that the number of UAVs charging at a charging station cannot exceed the number of charging slots there. Eqs. (6) and (7) are the service and flight constrains, respectively, to prohibit a UAV from serving and flying, respectively, in time slot t if that action would drain out all of its battery power. Constrain (8) limits a UAV to be in one of the four possible modes at any time. Eq. (9) allows a UAV to charge only in a subarea where a charging station resides and Eq. (10) disallows a UAV to enter standby mode in a subarea without any charging station. Eq. (11) is the mobility constraint, which specifies the set of subareas that a UAV can reach within a time slot duration.

Assume that UAVs are homogeneous in service capability (i.e., $R_p(a)$'s for all UAV p's are identical). We can form a undirected graph $G = (V, E)$ where V is the set of subareas and $(a, a') \in E$ if $a' \in R_p(a)$. If we want to statically place the fewest UAVs into the working area to meet the coverage demands of all subareas, the problem becomes the classic minimum multi-dominated set problem in G, which is NP-hard. Considering the fact that the static placement problem is only a special case of our problem, our problem is at least NP-hard.

IV. Proposed Approaches

We formulate the problem as a partially observable Markov decision process (POMDP) with multiple agents. We then details the proposed approaches to this problem.

A. Multi-Agent Partially Observable Markov Decision Process

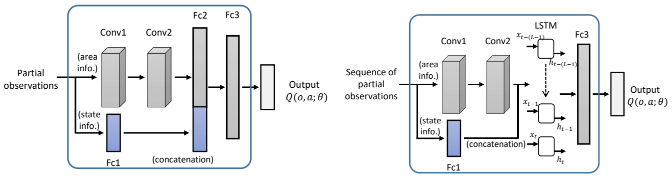
A multi-agent Markov Decision Process (MDP) consists of several essential elements. The first is the set of agents, which is P in our problem. Second, the set of actions of each agent. We classify agent's actions into four types: *fly*, *serve*, *charge*, and *rest*. An action actually comprises a sequence of agent's movements with a particular intention. The third element of a Markov game is the state space of each agent, where each state includes current status of all observable subareas, charging stations, and UAVs with respective locations relative to the observing agent. The use of relative (rather than absolute) coordinates makes the model independent of the size of the target area yet general enough for every agent to apply. It also significantly shrinks the state space. Another way to reduce the state space is to classify b_p^t, the residual energy of agent p in time slot t, into three levels, namely, high (H), medium (M) and low (L), which is denoted by B_p^t.

The last component is a reward fed by the environment to each agent after the agent takes an action. We design a mechanism to model the environment, which gives a positive or negative reward to each agent p depending on p's action and residual energy B_p^t in time slot t. For example, if p decides to serve in time slot t but $B_p^t = L$, p will receive a negative reward. If $B_p^t \in \{M, H\}$, p's reward for service depends on the number of subareas on which p's service is *effective* in time slot t. An agent's service on a subarea is effective if the number of all UAVs serving that subarea does not exceed the number demanded. In addition, an extra penalty (a negative reward) will be given to an agent when it drains out its battery, flies out of the boundary of the target area, or rests in a subarea where no charging station resides.

As UAVs have limited visibility, we define the observation space for each UAV. Let Sen_p^t be the set of subareas that are withing UAV p's visibility in time slot t. The observation space of p is defined as

$$obs_p^t = \left\{d'_a, s'_m \mid \forall a, m \in Sen_p^t\right\} \cup \left\{b_p^t, a_p^{t-1}, B_p^t\right\}, \quad (12)$$

where d'_a is the number of UAVs currently needed to meet the service demand of subarea a and s'_m is the number of currently



(a) DQN

(b) DRQN

Fig. 2: Model architecture

available charging slots in m. Note that p's observation does not include the status of any other UAV.

Our goal is to design a distributed mission and charging scheduling for rechargeable UAVs. We use DRL and follow the the paradigm of centralized training with decentralized execution. That is, all UAVs collaboratively train a common learning model offline, which is then used by each UAV to independently make on-line decisions in execution time.

B. Double-DQN (DDQN)

Deep Q-network (DQN) [18] is a model-free, temporal-difference, value-based, off-policy reinforcement learning method. It extends conventional Q-Learning by adding a deep neural network to non-linearly approximate the Q function used in Q-learning, i.e., $Q(s,a;\theta) \approx Q^*(s,a)$. The neural network finds the optimal weight θ by minimizing the following loss function $L(\theta)$ through gradient descent.

$$L(\theta) = \mathbf{E}\left[\left(r + \gamma \cdot \max_{a'} Q(s',a';\theta') - Q(s,a;\theta)\right)^2\right], \quad (13)$$

where $r + \gamma \cdot \max_{a'} Q(s',a';\theta')$ is target Q-value.

Fig. 2a shows our DQN architecture. We use two convolutional layers to extract geographic environment information. A fully connected layer is used to process observations plus UAV status information such as power level and previous action. The output of these layers are fed into two fully connected layers to get the final Q-value of each action.

We also use *experience replay* and *target network* to speed up the convergence rate of DQN. Experience replay accelerates the convergence speed of the neural network by breaking the temporal dependency relationship on Q-Learning algorithm. We use target network to update the evaluation network during training, and periodically update the target network. The use of target network can make the network to have better generalization ability and prevent overfitting. Algorithm 1 details the proposed DQN-based approach.

C. Deep Recurrent Q-Network (DRQN)

When using DQN, a UAV may be trapped in some working area where no service target is within its observable range. To this end, we propose Deep Recurrent Q-Network (DRQN) algorithm [19], which incorporates the DQN algorithm with the recurrent neural network (RNN). DRQN considers a series of past observations to learn continuous behaviors, which helps deciphering the underlying state of the POMDP.

Algorithm 1 DQN for multi-UAV scheduling

```
    Input: Number of UAVs —$P$—; working area —$A$—;
    Output: Q-value function $Q(o,a;\theta)$
 1: Initialize parameter setting
 2: Initialize Environment $env$
 3: Initialize primary network $Q_\theta$, target network $Q_{\theta^-}$, replay buffer $D$
 4: for each episode $e$ do
 5:     $t \leftarrow 0$
 6:     reset $env$
 7:     while $t \le T$ do
 8:         Sequentially select UAV agent $p$
 9:         Get observation $obs_p^t$ from UAV $p$
10:         Select random action $a_p^t$ with probability $\epsilon$
11:         otherwise $a_p^t = \arg\max_{a'} Q(obs_p^t, a'; \theta)$
12:         Execute action $a_p^t$ then get observation $obs_p^{t+1}$ and reward $r_p^t$
13:         Store transition $(obs_p^t, a_p^t, r_p^t, obs_p^{t+1})$ to $D$
14:         /* experience replay */
15:         Sample random mini-batch of transitions $(obs_q^j, a_q^j, r_q^j, obs_q^{j+1})$ from $D$
16:         Set $y = \begin{cases} r_q^j & \text{for terminal state} \\ r_q^j + \gamma \cdot \max_{a'} Q(obs_q^{j+1}, a'; \theta^-) & \text{for non-terminal state} \end{cases}$
17:         Do a gradient descent with loss $\left(y - Q(obs_q^j, a_q^j; \theta)\right)^2$
18:         /* periodic update of target network */
19:         Replace target parameters $\theta^- \leftarrow \theta$ every $N^-$ step
20:         if UAV $p$ drains out of energy then
21:             break
22:         end if
23:         if last UAV agent finishes then
24:             $t \leftarrow t + 1$
25:         end if
26:     end while
27: end for
```

For DRQN, we replace the FC2 layer in DQN with Long Short-Term Memory (LSTM) as shown in Fig. 2b. LSTM neurons pass hidden state and cell state messages in the temporal dimension, the former mainly stores short-term memory and the latter mainly stores long-term memory. We combine low-level features as the output of the convolution with the UAV information (past actions included) for each time slot in input sequence and feed them into the LSTM layer.

Because we incorporate LSTM layer in the algorithm, each UAV agent has to keep additional information about hidden state and cell state. The initial values of hidden state and cell state are all zeros. The UAV agent decides its action based on obs_p^t, hidden state, and cell state. We keep a sequence of past behaviors for each agent. The length of a sequence is 10, which is the farthest distance of the whole working area. The transition we collect is $(obs_p^{t-10}, \cdots, obs_p^t, a_p^t, r, o_p^{t+1})$.

The algorithm for DRQN is slightly different from Algorithm 1. First, a_p^t in Line 11 is set to $\arg\max_{a'} Q(obs_p^t), h_t, c_t, a'; \theta)$. Second, we store $(obs_p^{t-L-1}, \cdots, obs_p^t, a_p^t, r_p^t, obs_p^{t+1})$ to D in Line 13. Third, we sample random mini-batch of transitions $(obs_q^{j-L-1}, ..., obs_q^j, a_q^j, r_q^j, obs_q^{j+1})$ from D in Line 15. Fourth, θ^- in Line 16 is replaced with θ. Finally, the loss in Line 17 is $\left(y - Q(obs_q^{j+1}, a; \theta)\right)^2$.

Table I shows the parameters of the two proposed models.

TABLE I: Parameters of the models

DQN	Conv1 2x2, 8 BatchNorm Relu Conv2 2x2, 8 BatchNorm Relu Fc1 (6,30) Fc2 (422,128) Fc3 (128,9)	DRQN	Conv1 2x2, 8 BatchNorm Relu Conv2 2x2, 8 BatchNorm Relu Fc1 (6,30) LSTM (422,128) Fc3 (128,9)

TABLE II: Environment Setting

Parameter	Default Value	Range
Number of subareas (hexagonal shape)	100	-
Side length of the hexagon (m)	25	-
Whole area size (m^2)	162380	-
d_a: Coverage demand of each subarea	1	[0, 10]
Number of charging stations	5	-
s_m: Number of charging slots per station	9	[3, 9]

V. Simulation Results

We also compare our proposed approach to the game based approach proposed by Li et al. [4]. We considered a working area consisting of 100 hexagonal cells with side length 25 m. Each subarea demanded service from one UAV by default. Five charging stations were located, each consisted of 9 charging slots by default. Tables II to IV show the settings of the environment, the UAVs, and the training model, respectively.

A. Number of UAVs

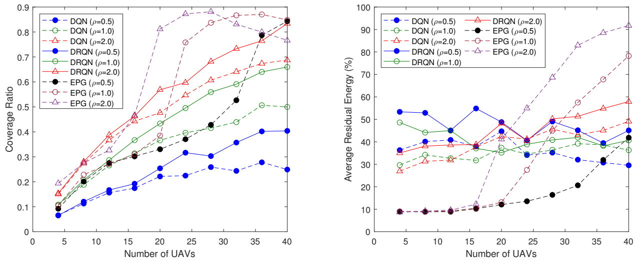
We varied the number of UAVs for a performance comparison between the proposed approaches and the EPG-based work by Li et al. [4]. Fig. 3a shows that the coverage ratio generally increased with the number of UAVs, which was expected. All the three approaches roughly performed the same when 20 or fewer UAVs were deployed. With those settings, there was little competition among UAVs for service coverage as UAVs were needed in most subareas. When more than 20 UAVs were deployed, the EPG-based approach achieved the highest coverage ratio due to the availability of complete information for UAVs to contend for service. When more UAVs than needed were deployed, the coverage ratio of the EPG-based approach turned to decline. On the other hand, the DRL-based approaches maintained an increasing trend as the number of UAVs increased.

TABLE III: UAV Setting

Parameter	Default Value	Range
Number of UAVs	20	[0, 40]
Range of UAV's service (m)	50	-
Range of UAV's observation (m)	200	-
Maximum energy budget $b_p^{\max}$ (kJ)	200	-
Energy consumption for hovering e_p^{hov} (kJ/time slot)	1	-
Energy consumption for serving e_p^{ser} (kJ/time slot)	1	-
ρ: Charging rate $k_p, \forall p \in P$ (kJ/time slot)	0.25	[0.25, 0.5, 1, 2]

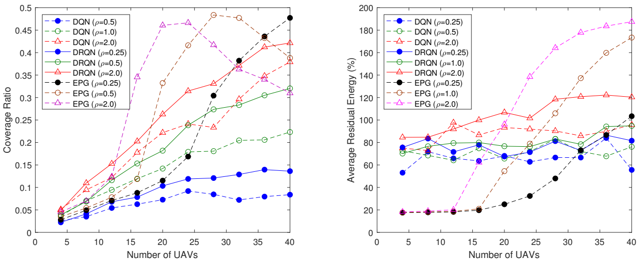
TABLE IV: Training Model Parameters

Parameter	Default Value	Range
Number of episodes	200	-
Maximum time slot per episode	2000	-
Batch size	256	-
Learning rate	0.005	-
Exploration rate of ϵ-greedy	0.1	-
Discount factor γ	0.9	[0, 1]
Replay memory capacity	3000	-
Target network iteration	100	-



(a) Coverage ratio (b) Average residual energy

Fig. 3: Results with increasing number of UAVs



(a) Coverage ratio (b) Average residual energy

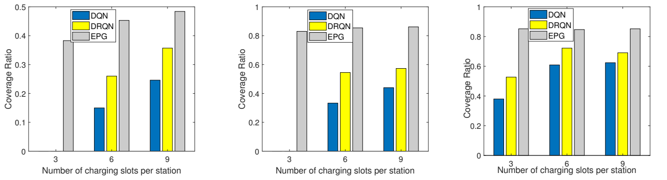
Fig. 4: Results with clustered distribution of service demands

Fig. 3b shows the average residual energy of the UAVs in operation. In the DQN and DRQN approaches, the number of UAVs had less impact on the average residual energy of the UAVs. This is because UAVs in our mechanisms do not share their observations and status. Even if some UAV realized that service was no longer needed in some area, other UAVs still spent time and energy in exploring that area. As a result, the average residual energy of the UAVs was between 25% and 55%.

The EPG approach had a low average residual energy when fewer than 20 UAVs were deployed. It actually traded energy for service coverage. When more UAVs were deployed, the average residual energy improved as UAVs spent more time in charging. After the coverage reached a peak, extra UAVs only increased the average residual energy but not the coverage ratio.

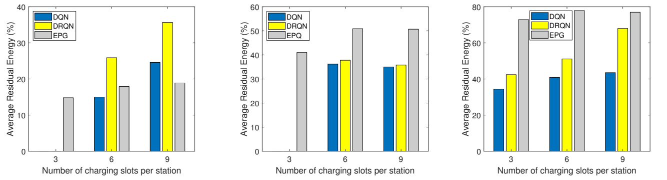
B. Clustered Distribution of Service Demands

We considered a scenario where the demands of UAV service were clustered together by locations, forming several hotspots in the target area. We used scikit-learn [20] to randomly pick up two hotspot centers and generate 100 coverage demands in the target area.



(a) $\rho = 0.5$ (b) $\rho = 1.0$ (c) $\rho = 2.0$

Fig. 5: Coverage ratios with various charging rates (ρ)



(a) $\rho = 0.5$ (b) $\rho = 1.0$ (c) $\rho = 2.0$

Fig. 6: Average residual energy with various charging rates (ρ)

Fig. 4a shows the result of coverage ratio with the clustered demand distribution, where the the values are roughly halved but the trend is similar compared against the result with a uniform demand distribution (Fig. 3a). Similar result can also be found in the case of residual energy (Fig. 4b).

C. Charging Capacity

We fixed the number of UAVs to 30 and varied the charging capacity by setting ρ to be 0.5, 1, and 2 and s_m to be 3, 6, and 9. Fig. 5 shows the coverage ratios.

When $\rho = 0.5$ or 1.0, the proposed approaches did not work with $s_m = 3$. A close examination reveals that when charging slots were fully occupied, UAVs in the proposed approaches did not learn to rest and wait at a charging station. Instead, they attempted exploring other charging stations due to limited observation. Such actions caused early terminations of the training episode when a UAV ran out of its battery power. The situation improved when charging slots were mostly available either because of using fast charging technology ($\rho = 2.0$) or more charging slots were set up ($s_m = 9$). EPG performed the best, followed by DRQN and then DQN.

Fig. 6 presents the results of average residual energy. The results also show that the residual energy improved with adequate charging capacities and fast charging rates.

VI. Conclusions

We have addressed the problem of distributed serving and charging scheduling for a UAV swarm to maximize service coverage intensity. We model the problem as a POMDP and propose two DRL-based approaches, one based on DQN and the other on DRQN. The experimental results show that the proposed approaches can handle distributed serving and charging scheduling as long as enough charging facility is provided. Although the proposed approaches are inferior to the counterpart in terms of coverage ratio and energy savings, they are more practical as UAVs are assumed to have limited visibility.

Acknowledgment

This work was supported by the National Science and Technology Council of Taiwan under grant number NSTC 112-2218-E-A49-023.

References

[1] H. Shakhatreh, A. Khreishah, J. Chakareski, H. B. Salameh, and I. Khalil, "On the continuous coverage problem for a swarm of UAVs," in *2016 IEEE 37th Sarnoff Symposium*, 2016, pp. 130–135.

[2] A. Trotta, M. Di Felice, F. Montori, K. R. Chowdhury, and L. Bononi, "Joint coverage, connectivity, and charging strategies for distributed UAV networks," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 883–900, Aug. 2018.

[3] L. Amorosi, L. Chiaraviglio, and J. Galán-Jiménez, "Optimal energy management of UAV-based cellular networks powered by solar panels and batteries: Formulation and solutions," *IEEE Access*, vol. 7, pp. 53 698–53 717, 2019.

[4] C.-I. Li, L.-H. Yen, and M.-C. Cho, "Distributed mission and charging scheduling for UAV swarm to maximize service coverage," in *Proc. IEEE VTC2021-Fall*, Sep. 2021.

[5] J. Hu, H. Zhang, and L. Song, "Reinforcement learning for decentralized trajectory design in cellular UAV networks with sense-and-send protocol," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6177–6189, Aug. 2019.

[6] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," *CoRR*, 2018.

[7] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3177–3192, Oct. 2021.

[8] R. Ding, F. Gao, and X. S. Shen, "3D UAV trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7796–7809, 2020.

[9] S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5994–6006, 2021.

[10] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of UAVs," in *IEEE Int'l Conf. on Autonomous Robot Systems and Competitions*, 2015, pp. 111–117.

[11] W. Zhang, Q. Wang, X. Liu, Y. Liu, and Y. Chen, "Three-dimension trajectory design for multi-UAV wireless network with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 600–612, 2021.

[12] L. Ruan, J. Wang, J. Chen, Y. Xu, Y. Yang, H. Jiang, Y. Zhang, and Y. Xu, "Energy-efficient multi-UAV coverage deployment in UAV networks: A game-theoretic framework," *China Commu.*, vol. 15, no. 10, pp. 194–209, 2018.

[13] L. Ruan, J. Chen, Q. Guo, H. Jiang, Y. Zhang, and D. Liu, "A coalition formation game approach for efficient cooperative multi-UAV deployment," *Applied Sciences*, vol. 8, p. 2427, 11 2018.

[14] H. Ghazzai, A. Kadri, M. Ben Ghorbel, H. Menouar, and Y. Massoud, "A generic spatiotemporal UAV scheduling framework for multi-event applications," *IEEE Access*, vol. 7, pp. 215–229, 2019.

[15] Y. Khosiawan, Y. Park, I. Moon, J. M. Nilakantan, and I. Nielsen, "Task scheduling system for UAV operations in indoor environment," *Neural Computing and Applications*, vol. 31, pp. 5431–5459, 2019.

[16] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu, "Energy efficient 3-D UAV control for persistent communication service and fairness: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 53 172–53 184, 2020.

[17] A. Trotta, M. D. Felice, F. Montori, K. R. Chowdhury, and L. Bononi, "Joint coverage, connectivity, and charging strategies for distributed UAV networks," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 883–900, 2018.

[18] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. the AAAI Conf. on Artificial Intelligence*, vol. 30, no. 1, 2016.

[19] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *2015 AAAI Fall Symposium Series*, 2015.

[20] "scikit-learn 1.0.1." [Online]. Available: https://scikit-learn.org/stable/modules/classes.html