Engineering Applications of Artificial Intelligence

Multi-UAV Trajectory Optimizer: A Sustainable System for Wireless Data Harvesting with Deep Reinforcement Learning

--Manuscript Draft--

Manuscript Number:	EAAI-22-4101
Article Type:	Research paper
Keywords:	Wireless sensor network, UAVs, Data harvesting, MARL, Transfer Learning, Scenario Parameters
Corresponding Author:	MinCheol Seong, B.S Republic of Korea Army Headquarters Daejeon, Yuseong-gu KOREA, REPUBLIC OF
First Author:	MinCheol Seong, B.S
Order of Authors:	MinCheol Seong, B.S
	Ohyun Jo, PhD.
	Kyungseop Shin, PhD.
Abstract:	The sea covers a large portion of the Earth, but most of the sea is not reachable by humans. Herein, a wireless sensor network consisting of multiple autonomous unmanned aerial vehicles (UAVs) is a promising solution to harvest data and understand the sea environment. However, the battery maintenance problem of sensor nodes and the complicated path planning problem of each UAV are still problematic. In this paper, we propose an optimal operation strategy based on multi-agent reinforcement learning (MARL) to tackle those hurdles. Various parameters such as the number of deployed UAVs, charging start capacity, and charging complete capacity define a multi-UAV system. This approach is applicable without a time-consuming and costly policy control. We also describe how to balance multiple objectives such as data harvesting, charging, and collision avoidance using transfer learning. Finally, learning a policy control that generalizes multiple scenario parameters allows us to analyze the performance of individual parameters in a specific scenario, which helps to find the macro-level optimal parameter within a particular scenario.
Suggested Reviewers:	Marco Caccamo, PhD Professor, Technical University of Munich mcaccamo@tum.de He has been actively researching using reinforcement learning in UAV-aid networks.
	Harald Bayerlein, PhD A postdoctoral researcher, Technical University of Munich h.bayerlein@tum.de He has been actively researching using reinforcement learning in UAV-aid networks.

Multi-UAV Trajectory Optimizer: A Sustainable System for Wireless Data Harvesting with Deep Reinforcement Learning

Mincheol Seong[1], Ohyun Jo[2] and Kyungseop Shin[3]

***Abstract*—The sea covers a large portion of the Earth, but most of the sea is not reachable by humans. Herein, a wireless sensor network consisting of multiple autonomous unmanned aerial vehicles (UAVs) is a promising solution to harvest data and understand the sea environment. However, the battery maintenance problem of sensor nodes and the complicated path planning problem of each UAV are still problematic. In this paper, we propose an optimal operation strategy based on multi-agent reinforcement learning (MARL) to tackle those hurdles. Various parameters such as the number of deployed UAVs, charging start capacity, and charging complete capacity define a multi-UAV system. This approach is applicable without a time-consuming and costly policy control. We also describe how to balance multiple objectives such as data harvesting, charging, and collision avoidance using transfer learning. Finally, learning a policy control that generalizes multiple scenario parameters allows us to analyze the performance of individual parameters in a specific scenario, which helps to find the macro-level optimal parameter within a particular scenario.**

***Index Terms*—Wireless sensor network, UAVs, Data harvesting, MARL, Transfer Learning, Scenario Parameters**

I. Introduction

THERE are many fields where data harvesting applications are needed, one of which is the sea. The sea, which covers more than 70% of the Earth, is essential for life and has been gaining considerable attention recently for various applications in the marine industry, climate, oceanic disaster prevention, and military. For realization, it is essential to continuously monitor changes in the sea environment, such as climate change or contaminant detection. Many studies have been conducted for dynamic information monitoring systems in the sea environment through wireless sensor nodes [1], [2]. By the way, it is still challenging for sensor nodes to collect the data continuously produced in the sea environments due to the limited transmission energy and the long distance from a receiver located on land. Autonomous unmanned aerial vehicles (UAVs) as data harvesters can be one of the promising solutions to resolve these problems. They have many advantages over satellites and ground base stations [3]. A quick and straightforward deployment would cover a wide range of areas and reduce the cost of the infrastructure installed on the land. In addition, if UAVs fly close to the sensor nodes, the energy problem can be relaxed. Then, it will result in an increase in the sustainability and throughput efficiency of sensor nodes since the main technological issue related to sensor nodes in the sea environments is energy conservation [4].

In this work, we focused on building the UAV-aided data collection system based on a deep reinforcement learning mechanism to maximize sustainability. A team of UAVs consists of a variable number of homogeneous drones with the same task of collecting data. While carrying out the tasks, the UAVs need to return to the charging station when the battery is getting low. This issue imposes complex constraints on the design of the UAV trajectory for optimal data collection with limited battery capacity. Moreover, collision avoidance among UAVs should be considered to optimize network efficiency when the number of UAVs increases.

In this regard, Deep Reinforcement Learning (DRL) might provide an opportunity to alleviate the challenges to our goals. First, it can be applied to complex problems with many constraints through an intuitive combination of reward functions. In addition, since the UAV control problems in the communication scenario have been proven to be NP-hard in many cases [5], [6], and the DRL is computationally efficient for approaching such kind of NP-hard problem as shown in [7].

A. Related Works

Multi-UAVs have recently been actively studied for their flexible and efficient operations in various fields and communications and networks [8].

In [9], the authors suggested a method for a UAV to collect data through wireless charging. They adopted a single agent approach using Q-learning. However, it is impossible to consider all the complex constraints and requirements in reality with the single agent approach. There have been studies on approaches using DRL in UAV-aided networks to describe the complex environment in reality [10], [11]. They considered convolutional neural networks and partial observability to define the environment as a partially observable Markov decision process (POMDP) for multi-UAV path planning. In a vast scenario parameter space in which environmental parameters are changed, there is no need for a separate relearning to be introduced in [12]. Although it is possible to measure the performance of various scenario parameters, there is no consideration for multiple objectives.

[1]M. Seong is with the Republic of Korea Army, Pocheon, Republic of Korea (email: smc5741@gmail.com).

[2]O. Jo is with the Department of Computer Science, Chungbuk National University, Cheongju, 28644, Republic of Korea(email: ohyunjo@chungbuk.ac.kr).

[3]K. Shin is with the Department of Computer Science, Sangmyung University, Seoul, Republic of Korea (email: ksshin@smu.ac.kr).

The multiple objectives of the DRL approach are studied in [13], where Deep Q-Network (DQN) and transfer learning are leveraged in a multi-agent setting. They tried to achieve multiple objectives for autonomous vehicles to fulfill customer demands for charging. However, to optimize the charging system, it needs to consider realistic environments.

We focus on constructing a sustainable UAV-aided network system by training UAVs with multiple objectives, including data collection, navigation constraints, and charging in a dynamic sea environment. To the best of the authors' knowledge, this study is the first to address multi-UAV path planning based on the collective combination of a multi-agent DRL and a transfer learning in which the learned control policies can achieve multiple objectives.

B. Organizations

The rest of the paper is organized as follows.

In Section II, we describe the system model to present the UAV, network environment, and channel. All the system parameters are reflected into the POMDP to consider our practical DRL application scenarios in Section III. Multi-agent DRL learning approach and transfer learning to optimize the system are introduced in Section IV. Simulation results are presented and analyzed in Section V. Finally, the conclusion and future work are given in Section VI.

II. System Model

A. Environmental Model

In this section, a system model is addressed to make the RL approach applicable to the real world's dynamics. The architecture and the parameters are designed to apply the RL method to our environmental model.

The first step required to apply the RL model is to create an environmental model that reflects the behavior of the agents which is likely to occur in real environments. It is needed to consider the places where UAVs can take off and charge and sensing points where sensor nodes are placed. An example of a grid world is illustrated in Fig. 1, and a single UAV trajectory are marked as described in the attached legend in Tab. I.

TABLE I: Legend for scenario plots.

	Symbol	Description
DQN	■	Start and charging zone
Input	●	Sensor node
Visualization	◇	Starting positions during an episode
	◆	Termination of an episode
	→	UAV movement while comm. with red device
	×	Hovering while comm. with red device
	★	Hovering while charging

B. UAV Model

The UAVs are placed and move within the grid world with a size of $M \times M$. The state of UAV i at time t $\boldsymbol{s}_i(t)$ includes the following three elements:

1) position $\boldsymbol{p}_i(t) = [x_i(t), y_i(t), z_i(t)]^T \in \mathbb{R}^3$, where altitude $z_i(t) \in \{h\}$ is always a constant altitude h;

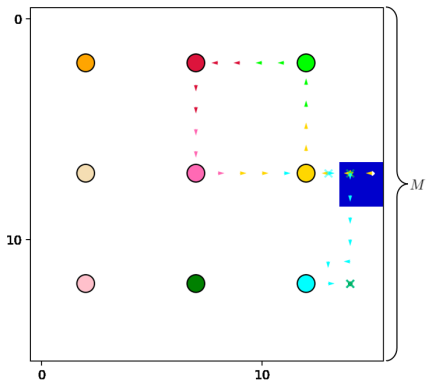


Fig. 1: Example of a single UAV collecting data from sensor nodes at the sea surface and charging when necessary in a sea environment of size 16×16 with a start and charging zone. Colored circles refer to sensor nodes placed at regular intervals floating at the sea surface. Different colors for each node make it easy to figure out which sensor node the UAV is communicating with.

2) operational status $\boldsymbol{\phi}_i(t) \in \{0, 1\}$, either inactive or active;
3) battery energy level (%) $\boldsymbol{b}_i(t) \in \mathbb{N}$.

Total operation time is set to $\boldsymbol{T}$ time steps for all UAVs. A mission time step with length δ_t is divided into m communication time steps. When the UAV starts charging or completes charging, $\boldsymbol{b}_i(t)$ is defined as charging start capacity or charging complete capacity, respectively, and is described as follows:

1) charging start capacity (%) $\mathbf{C}_S$;
2) charging complete capacity (%) $\mathbf{C}_C$.

For instance, $\mathbf{C}_S$ = 20 means that the UAV can charge its energy from 20% of the full battery capacity, and $\mathbf{C}_C$=70 means that the UAV charges up to 70% of the full battery capacity. Here, we define the relation between $\mathbf{C}_S$ and $\mathbf{C}_C$ as

$$10 \leq \mathbf{C}_S < \mathbf{C}_C \leq 100, \tag{1}$$

where $\mathbf{C}_C$ is always greater than $\mathbf{C}_S$, and UAVs cannot charge their energy beyond 100% of the entire battery capacity.

Each UAV is authorized to choose their action from action spaces, $\mathbf{A} = \{North,\ East,\ South,\ West,\ Hover\}$. While the UAV is charging or counters with either other UAVs or grid world boundaries, the required action of the UAV is $Hover$. Each UAV's physical action $\mathbf{a}_i(t) \in \widetilde{\mathbf{A}}(\boldsymbol{s}_i(t))$ can be limited to (2) if a charging condition $\mathbf{C}_i(t)$ is satisfied,

$$\widetilde{\mathbf{A}}(\boldsymbol{s}_i(t)) = \begin{cases} Hover, & \mathbf{C}_i(t) = 1 \wedge\ \boldsymbol{b}_i(t) \leq \mathbf{C}_C \\ \mathbf{A}, & otherwise, \end{cases} \tag{2}$$

which means that once the UAVs start charging, they have to keep charging until $\boldsymbol{b}_i(t) = \mathbf{C}_C$.

The charging determination $\iota_i(t)$ is required to start charging when $\boldsymbol{b}_i(t)$ is lower than $\mathbf{C}_S$ and to charge the battery until it reaches the $\mathbf{C}_C$.

$$\iota_i(t) = \begin{cases} 1, & \boldsymbol{b}_i(t^p) \leq \mathbf{C}_S \\ 0, & \boldsymbol{b}_i(t) > \mathbf{C}_C, \end{cases} \tag{3}$$

where t^p is a fixed prior time slot before t. Note that the default value of $\iota_i(t)$ at the beginning of an episode is 0.

The charging condition $\mathbf{C}_i(t)$ is defined as follows in which i-th UAV start to charge their energy only when the following conditions are satisfied.

$$\mathbf{C}_i(t) = \begin{cases} 1, & \boldsymbol{p}_i(t) \in \mathbf{L} \\ & \wedge\ \iota_i(t) = 1 \\ & \wedge\ \boldsymbol{a}_i(t) = Hover \\ 0, & otherwise. \end{cases} \tag{4}$$

The maximum moving displacement a UAV can move in one mission time slot is the grid cell size c. The speed of UAVs $\mathbf{v}_i(t)$ is constant within a time slot by dividing enough communication time slots. $\mathbf{v}_i(t)$ is upper bounded to moving at a horizontal velocity $V = c/\delta_t$ or hovering, i.e. $\mathbf{v}_i(t) \in \{0, V\}$ for all $t \in [0, T]$.

The position of the UAV is updated with the model given as

$$\boldsymbol{p}_i(t+1) = \begin{cases} \boldsymbol{p}_i(t) + \boldsymbol{a}_i(t), & \phi_i(t) = 1 \\ \boldsymbol{p}_i(t), & otherwise, \end{cases} \tag{5}$$

making the UAV update only if active.

A communication quota of each UAV $\mathbf{U}_i(t) \in \{1, 0\}$ indicates communication is either complete or incomplete, which can only be achieved through at least one charge. Note that the predefined goal of communication quota $\mathbf{U}_C \in \mathbb{N}$ satisfies the following relation,

$$m \times \boldsymbol{b}_i(0) < \mathbf{U}_C, \tag{6}$$

where $m \in \mathbb{N}$ is used to divide a mission time slot into m communication time slots (see II-C). If UAVs take off, $\mathbf{U}_i(t)$ is incomplete, and $\mathbf{U}_i(t)$ is complete when the cumulative number of communications of the agent reaches $\mathbf{U}_C$.

The evolution of the operational status $\boldsymbol{\phi}_i(t)$ of each UAV is described as

$$\boldsymbol{\phi}_i(t+1) = \begin{cases} 0, & \mathbf{U}_i(t+1) = 1 \\ & \vee\ \boldsymbol{\phi}_i(t) = 0 \\ & \vee\ b_i(t+1) = 0 \\ 1, & otherwise, \end{cases} \tag{7}$$

where if UAVs take off, $\boldsymbol{\phi}_i(t)$ is always active, and the operational status $\boldsymbol{\phi}_i(t)$ is inactivated when the communication quota $\mathbf{U}_i(t)$ is completed or the battery energy is depleted.

The i-th UAV's battery energy level updates according to

$$\boldsymbol{b}_i(t+1) = \begin{cases} \boldsymbol{b}_i(t) - 1, & \phi_i(t) = 1\ \wedge\ \mathbf{C}_i(t) = 0 \\ \boldsymbol{b}_i(t) + 10, & \phi_i(t) = 1\ \wedge\ \mathbf{C}_i(t) = 1 \\ \boldsymbol{b}_i(t), & otherwise, \end{cases} \tag{8}$$

where the battery energy level of each UAV $\boldsymbol{b}_i(t)$ is continuously decreased by 1 at each time slot when the operational status is active. Since the battery continues to decrease by one according to the time slot, remaining battery capacity means remaining flying time. UAVs can charge their battery energy level by ten at each time slot when charging.

C. Communication Channel Model

To describe communication between UAVs and sensor nodes, we introduce the concept of a communication time slot $n \in [0, \boldsymbol{N}]$ with $\boldsymbol{N} = m\boldsymbol{T}$ that is different from the mission time slot. Each mission time slot is divided into enough communication time slots so that the position and channel gain of the UAVs within one communication time slot are assumed to be constant. One communication time at slot n has length of $\delta_n = \delta_t/m$ minutes.

The k-th sensor node is located at the sea surface with $\mathbf{N}_k = \left[x_k, y_k, 0\right]^{\mathbf{T}} \in \mathbb{R}^3$ with $k \in \mathrm{K}$ where K is the total number of sensor nodes, which have stable locations during entire mission time slot $\boldsymbol{T}$ and the same distance between sensor nodes with fixed coordinates. At the start of the mission, a certain range of random data $\mathbf{D}_{k,init}$ is assigned to the k-th sensor node, and as dynamic sea environment information continues to generate data after the start of the mission, we define that the k-th sensor node generates new data $\mathbf{D}_{k,new} \sim Poisson(\lambda_p)$ according to the Poisson distribution.

A UAV-to-ground channel model [14] is employed to analyze the communication between UAVs and sensor nodes. The maximum achievable data rate at communication time slot n for the k-th sensor node is defined as

$$R_{i,k}(n) = \delta_n \log_2(1 + \mathrm{SNR}_{i,k}(n)). \tag{9}$$

Note that the length of a communication time slot is multiplied so that the maximum achievable data rate within a mission time slot is constant regardless of the number of communications.

The $\mathrm{SNR}_{i,k}(n)$ is derived as

$$\mathrm{SNR}_{i,k}(n) = \frac{P_{i,k}}{\sigma^2} \cdot d_{i,k}^{-\alpha_e} \cdot 10^{\eta_e/10}, \tag{10}$$

where P_k is transmit power of sensor node k, σ_e^2 is white Gaussian noise power at the receiver, $d_{i,k}$ is the distance between UAV i and sensor node k, α_e is path loss exponent and $\eta_e \sim \boldsymbol{N}(0, \sigma_e^2)$ is a Gaussian random variable. Since the sea environment is a Line of Sight (LOS) point-to-point communication because there are no obstacles, propagation parameter $e \in \{\mathrm{LoS}\}$ is defined.

For communication between sensor nodes and a UAV, we use the time-division multiple access (TDMA) model as a multiple access protocol. To utilize the models we defined earlier, three significant assumptions are given without loss of generality. First, it is assumed that the communication channel between a UAV and the sensor nodes works as an orthogonal resource block with the communication channel between the other UAVs and the sensor nodes. Eventually, there is no inter-UAV interference. Also, assuming that sensor nodes are in multi-band mode, a sensor node can communicate with multiple UAVs simultaneously, which means that scheduling decisions can be ignored when determining the action space of the UAVs. Finally, assuming UAVs always communicate according to the maximum rate, the sensor node only communicates with at most one UAV in a communication time slot. Based on the maximum rate constraint, the scheduling variable $q_{i,k}(n) \in \{0,1\}$ is chosen for all UAVs and all sensor

nodes in communication time slot n, in which $q_{i,k}(n)$ is equal to 1 only for the highest $\mathrm{SNR}_{i,k}(n)$.

Then, we can define the k-th sensor node's residual data within communication time slot n according to

$$d_k(n+1) = d_k(n) - \sum_{i=1}^{\mathrm{I}} q_{i,k}(n) R_{i,k}(n) \delta_n, \tag{11}$$

where I means the total number of agents. The residual data $\mathbf{D}_k(t)$ of the k-th sensor node at mission time slot t is as follows.

$$\mathbf{D}_k(t) = \begin{cases} \mathbf{D}_{k,init}, & t = 0 \\ d_k(n_t) + \mathbf{D}_{k,new}, & otherwise, \end{cases} \tag{12}$$

where n_t is the communication time slot at mission time slot t. Also, we assume that new data $\mathbf{D}_{k,new}$ is generated at each mission time slot not each communication time slot.

III. MARKOV DECISION PROCESS (DEC-POMDP)

The aforementioned operation for the UAV agents and sensor nodes is analyzed as a decentralized partially observable Markov decision process (Dec-POMDP) [15], which is defined as a tuple $(\mathbb{S}, \mathbb{A}, T, \mathbb{O}, O, R, \gamma)$ for practical application in RL. $\mathbb{S}$ stands for the state space, $\mathbb{A}$ is the joint action space, and T contains the transition probability functions. $\mathbb{O}$ means the joint observation space, O is the observation probability function that converts state information into unique observations for one agent. R contains the reward functions. The discount factor γ determines whether to weigh long-term or short-term rewards.

A. State Space

The state-space of the multi-agent problem consists of environment information, the state of the agents, and the state of the sensor nodes. It is defined as

$$\begin{aligned} \mathbf{S} = & \underbrace{\mathbf{L}}_{\substack{Start\ and \\ Charging\ zone}} && \} Environment \\ & \times \underbrace{\mathbb{R}^{\mathrm{I}\times 3}}_{\substack{UAV \\ Positions}} \times \underbrace{\mathbb{N}^{\mathrm{I}}}_{\substack{Flying \\ Times}} \times \underbrace{\mathbb{B}^{\mathrm{I}}}_{\substack{Operational \\ Status}} && \} UAV\,Agents \\ & \times \underbrace{\mathbb{R}^{\mathrm{K}\times 3}}_{\substack{Sensor\ Node \\ Positions}} \times \underbrace{\mathbb{R}^{\mathrm{K}}}_{\substack{Sensor\ Node \\ Residual\ Data}}. && \} Sensor\ Nodes \end{aligned}$$

The elements $s_i(t) \in \mathbf{S}$ are

$$\begin{aligned} s_i(t) = (\mathbf{H}_{layer}, \{\boldsymbol{p}_i(t)\}, \{\boldsymbol{b}_i(t)\}, \{\boldsymbol{\phi}_i(t)\}, \\ \{\mathbf{N}_k\}, \{\mathbf{D}_k(t)\}), \ \forall i \in \mathbb{I}, \ \forall k \in \mathbb{K}. \end{aligned} \tag{13}$$

$\mathbf{H}_{layer} \in \mathbb{B}^{H\times H\times 1}$ is the tensor layer of the start and charging zone $\mathbf{L}$. The other parts of the tuple are positions, remaining flying times, the operational status of all agents, and positions and residual data that can be collected from all sensor nodes.

B. Safety Controller

A safety controller is introduced to control UAV agents to operate within the boundaries of the grid world and avoid collisions with each other [12]. The safety controller alerts agents to cancel the planned action and to hover in the prior position when the agent's position in the next time slot reaches the boundaries of the grid world or is located at the same position as other agents. The actual action to be executed by the i-th agent through the safety controller is called a safety action $\boldsymbol{a}_i^s(t)$.

C. Reward Function

The operational order of the agents is as follows: advance each agent simultaneously, resolve their boundary counters or collisions with each agent, and consume their energy levels. As a result, the agents decide whether to charge their energy or communicate with the sensor node with the best data rate. Finally, new data is generated for each sensor node. Each of these transitions will have a reward associated with each agent. The reward function is comprised of the following elements:

$$\begin{aligned} r_i(t) = \alpha \sum_{k\in\mathrm{K}} (\mathbf{D}_k(t+1) - \mathbf{D}_k(t)) + \beta_i(t) \\ + \ \nu_i(t) + \ c_i(t) + \ \omega_i(t) + \ \zeta. \end{aligned} \tag{14}$$

The first term of the sum is a collective reward for all agents' collected data from all sensor nodes within a mission time slot t. It is weighted through the data collection multiplier α. The second term is an individual penalty when agents collide with the boundaries of the grid world or other agents and are given through

$$\beta_i(t) = \begin{cases} \beta, & \boldsymbol{a}_i(t) \neq \boldsymbol{a}_i^s(t) \\ 0, & otherwise, \end{cases} \tag{15}$$

where the penalty is characterized through the safety penalty value β. The third term $\nu_i(t)$ is the individual penalty for not fully communicating until it runs out of energy given by

$$\nu_i(t) = \begin{cases} \nu, & \boldsymbol{b}_i(t+1) = 0 \ \wedge \ \mathbf{U}_i(t+1) = 0 \\ 0, & otherwise, \end{cases} \tag{16}$$

The fourth term is a small positive reward while charging and given as

$$c_i(t) = \begin{cases} c, & \mathbf{C}_i(t+1) = 1 \ \wedge \ \phi_i(t+1) = 1 \\ 0, & otherwise, \end{cases} \tag{17}$$

where the reward c is utilized to encourage charging. The fifth term is a positive reward given when the communication quota is fulfilled and defined as

$$\omega_i(t) = \begin{cases} \omega, & \mathbf{U}_i(t+1) = 1 \\ 0, & otherwise, \end{cases} \tag{18}$$

parameterized through the reward ω to encourage charging. We found that simply imposing a large negative penalty ν when agents run out of their energy is not enough to balance the two main objectives, data collecting and charging. We were able to achieve two objectives in a balanced way with the small positive reward c given while charging and the relatively

large positive reward ω given when the communication quota was filled as well as transfer learning. The last term ζ is a constant movement penalty, which induces the agents to find more efficient trajectories.

D. Observation Space

We need to change the state space $\mathbf{S}$ to the observation space for POMDP. As shown in [16], we define an observation space for sensor node data, UAV flying times, and operational status through map processing algorithms that consist of map centering, which is the first step, and global-local map processing, which is the second step. Map centering is to place the agent in the center of the map to improve the agent's learning performance. Although there is a disadvantage of increasing the size of the map that results in increased size of neural networks to be trained with several trainable parameters, there is an advantage that the agent's action can be planned only based on the relative position to features, e.g., distance to start and charging zone, according to the agent's current position. After map centering, there are relatively distant features for which the agents should consider comprehensive path determination and relatively close features such as collision avoidance for path planning. The second step is to ignore some of the details of the distant features, significantly reducing the size of the neural network to be trained while maintaining the training performance shown in [16].

1) Mapping: The map centering and global-local algorithms can be used after the state spaces are stacked in the form of a map layer that projects the relative features according to the agent's position. According to the positions of the agents and sensor nodes, mapping is performed in pairs as follows:

$$(\{\boldsymbol{p}_i(t)\},\{\boldsymbol{b}_i(t)\}),(\{\boldsymbol{p}_i(t)\},\{\boldsymbol{\phi}_i(t)\}),(\{\mathbf{N}_k\},\{\mathbf{D}_k(t)\}),$$

where the size of each mapped map layer is $\mathbf{A}_{layer} \in \mathbb{R}^{M\times M}$ and the map layer $\mathbf{A}_{layer}$ can be stacked as a tensor of $\mathbb{R}^{M\times M\times n}$ when each layer has the same type.

2) Map centering and Global-Local Map: The map layer $\mathbf{A}_{layer} \in \mathbb{R}^{M\times M\times n}$ is converted into a centered tensor $\mathbf{B}_{layer} \in \mathbb{R}^{M_c\times M_c\times n}$ with $M_c = 2M-1$. In the centered tensor $\mathbf{B}_{layer}$, the original tensor $\mathbf{A}_{layer}$ is moved so that position of the agent is centered, and the rest of the extended part of the centered tensor $\mathbf{B}_{layer}$ is filled with the padding value $\mathbf{x_{pad}} \in \mathbb{R}$.

The tensor $\mathbf{B}_{layer} \in \mathbb{R}^{M_c\times M_c\times n}$ is addressed in two ways. In order to create a local map, the centered tensor $\mathbf{B}_{layer}$ is cropped with cropping parameter l. A local tensor $\mathbf{C}_{layer}$ which is the expression of local map is appeared as $\mathbf{C}_{layer} \in \mathbb{R}^{l\times l\times n}$. For a global tensor $\mathbf{D}_{layer}$ which is the expression of a global map, we use an average pooling algorithm with pooling cell size g to ignore the detail level of distant features. A global tensor $\mathbf{D}_{layer}$ is appeared as $\mathbf{D}_{layer} \in \mathbb{R}^{\lfloor \frac{M_c}{g} \rfloor \times \lfloor \frac{M_c}{g} \rfloor \times n}$. As l increases, the size of the local map increases, while as g increases, the pooling cell size increases, reducing the size of the global map.

3) Observation Space: Through the map processing, observation space $\mathbf{\Omega}$, which is the input space for the agent, is defined as

$$\mathbf{\Omega} = \underbrace{\mathbf{\Omega}_l}_{\text{Local Map}} \times \underbrace{\mathbf{\Omega}_g}_{\text{Global Map}} \times \underbrace{\mathbb{N}}_{\text{Flying Time}}, \tag{19}$$

where the local map is

$$\mathbf{\Omega}_l = \underbrace{\mathbb{B}^{l\times l\times 1}}_{\substack{Local\\ Environment}} \times \underbrace{\mathbb{R}^{l\times l}}_{\substack{Local\\ Data}} \times \underbrace{\mathbb{N}^{l\times l}}_{\substack{Local\\ Flying\ time}} \times \underbrace{\mathbb{B}^{l\times l}}_{\substack{Local\\ Operat'l\ status}}, \tag{20}$$

and the global map is

$$\mathbf{\Omega}_g = \underbrace{\mathbb{R}^{\tilde{g}\times\tilde{g}\times 1}}_{\substack{Global\\ Environment}} \times \underbrace{\mathbb{R}^{\tilde{g}\times\tilde{g}}}_{\substack{Global\\ Data}} \times \underbrace{\mathbb{R}^{\tilde{g}\times\tilde{g}}}_{\substack{Global\\ Flying\ time}} \times \underbrace{\mathbb{R}^{\tilde{g}\times\tilde{g}}}_{\substack{Global\\ Operat'l\ status}}, \tag{21}$$

with $\tilde{g} = \left\lfloor \frac{M_c}{g} \right\rfloor$. Observation space $o_i(t) \in \mathbf{\Omega}$ is defined as a tuple

$$\begin{aligned} o_i(t) = (&\mathbf{M}_{l,i}(t), \mathbf{D}_{l,i}(t), \mathbf{B}_{l,i}(t), \mathbf{\Phi}_{l,i}(t), \\ &\mathbf{M}_{g,i}(t), \mathbf{D}_{g,i}(t), \mathbf{B}_{g,i}(t), \mathbf{\Phi}_{g,i}(t), b_i(t)). \end{aligned} \tag{22}$$

In an observation space tuple, $\mathbf{M}_{l,i}(t)$ is the local observation of agent i of the environment, $\mathbf{D}_{l,i}(t)$ is the local observation of the residual data, $\mathbf{B}_{l,i}(t)$ is the local observation of the remaining flying time, and $\mathbf{\Phi}_{l,i}(t)$ is the local observation of the operational status. $\mathbf{M}_{g,i}(t), \mathbf{D}_{g,i}(t), \mathbf{B}_{g,i}(t)$, and $\mathbf{\Phi}_{g,i}(t)$ are the respective global observations. Note that surplus elements of the remaining flying time allow the agents to understand their remaining flying time better.

As a result, we artificially transformed the POMDP to solve the operational systems problem we defined and significantly reduced the neural network's size as proved in [16], making the training procedure feasible and reducing the training time.

IV. Multi-agent Reinforcement Learning(MARL) and Transfer Learning

Now we can formulate the RL as Dec-POMDP is depicted (see III-D). RL is a policy optimization process that selects an action a_i based on a policy π in the current state s_i and receives feedback called a reward r_i through interaction with the environment. At the same time, it changes to the next state s_i'. The basic framework for RL is shown in Fig. 2. Note that, because of the Dec-POMDP we defined, the state that becomes the input space for the agent is an observation state o_i observed by the agent, not the actual state s_i.

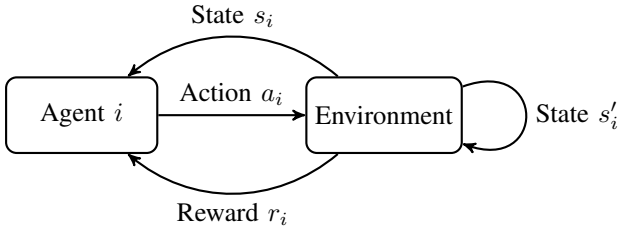


Fig. 2: The basic framework of RL.

A. Double Deep Q-learning

One of the promising algorithms in RL is Q-learning [17], which maximizes the cumulative discounted reward by repeatedly acting a_i in state s_i until the end of the episode. It is given as

$$Q(s_i, a_i) = r(s_i, a_i) + \gamma_q \sum_{s'_i \in S} p(s_i, a_i, s'_i) Q^*(s'_i, a'_i). \quad (23)$$

Note that $s_i,\ a_i$ is the same as $s_i(t),\ a_i(t)$ as well as $s'_i,\ a'_i$ is the same as $s_i(t+1),\ a_i(t+1)$ for the ease of exposition. In (23), $r(s_i, a_i)$ represents the immediate reward given action a_i in state s_i, $\gamma_q \in [0, 1]$ is the discounted factor, $p(s_i, a_i, s'_i)$ is the transition probability to next state s'_i after action a_i in the present state s_i, and $Q^*(s'_i, a'_i) = \max Q(s'_i, a'_i)$ represents the optimal Q-value of action a'_i in next state s'_i. Q-value, which is the expected value of the sum of future rewards given the state s_i, action a_i, is updated by the following formula:

$$Q(s_i, a_i) \leftarrow (1 - \alpha_q) Q(s_i, a_i) + \alpha_q \times \left[r(s_i, a_i) + \gamma_q \sum_{s'_i \in S} p(s_i, a_i, s'_i) Q^*(s'_i, a'_i) \right], \quad (24)$$

where $\alpha_q \in (0, 1]$ is a learning rate. As such, the optimal policy π_* is derived from

$$\pi_*(s_i) = \underset{a_i \in A}{argmax}\ Q(s_i, a_i). \quad (25)$$

Q-learning is performed using a Q-table where rows and columns consist of action and state to calculate the Q-value. Q-learning is computationally effective when the environment is simple. Still, when the environment becomes more complex with highly extensible state space, it becomes challenging to converge because the Q-table grows exponentially.

DQN was introduced to approximate the Q-table using deep neural networks with weights θ [18]. The advantage of function approximation is to converge DQN in complex environments where neural networks show significant data efficiency in generalization with experience replay and the target network. Experience replay allows networks to store transitions (s_i, a_i, r_i, s'_i) in the experience replay memory $\mathbf{D}$. During training, instead of deploying only the current transition (s_i, a_i, r_i, s'_i), mini-batches of transitions are chosen from the replay memory and used to train the network, which leads to reduce correlations in the sequence of training transitions as well as to prevent to be driven to a local minimum. The target network is a separate network that duplicates the original network with weights θ^-. This network is updated at slower intervals than the original network to stabilize the overall convergence of the network during training. Then, the Q-network updates its weights to minimize a loss function represented by the mean square error as such.

$$L_i(\theta) = \mathbb{E}_{s_i, a_i, r_i, s'_i \sim \mathbf{D}} \left[|| y_i^{DQN} - Q_i(s_i, a_i; \theta) ||^2 \right], \quad (26)$$

where the target value is

$$y_i^{DQN} = r_i(s_i, a_i) + \gamma_q \max_{a'_i \in A_i} Q_i(s'_i, a'_i; \theta^-). \quad (27)$$

The parameters of the target network θ^- are updated either with a periodic simple copy of θ for each episode or a soft update for each state transition we utilized as the target network update method

$$\theta^- = \theta\tau + \theta^-(1 - \tau), \quad (28)$$

where $\tau \in [0, 1]$ is an update factor and we choose τ as 0.005.

The target value y_i^{DQN} is inevitably different from the optimal value Q^* because the action of the next state a'_i is selected with the target network parameter θ^- and then updated to the max value of $Q_i(s'_i, a'_i; \theta^-)$ with the same target network parameter θ^-. The overestimation problem occurs because the max calculation is performed twice with the same parameter θ^-. Double-deep Q networks (DDQN), which separate the parameters of the Q network when calculating the target value y_i^{DDQN}, is introduced [19]. Finally, the target value y_i^{DDQN} is defined as

$$y_i^{DDQN} = r_i(s_i, a_i) + \gamma_q Q_i \left(s'_i, \underset{a'_i \in A_i}{\text{argmax}} Q_i(s'_i, a'_i; \theta); \theta^- \right). \quad (29)$$

B. Multi-Agent Deep RL (MADRL) and Transfer Learning

There are a few approaches to address a challenge in optimizing each agent's policy control while considering the other agents in the multi-agent environment. The best way to apply our learning approach is decentralized deployment with a knowledge transfer [20]. With this approach, agent execution occurs individually, which is the decentralized selection of the policy. We chose all agents to train and learn their policies on the same network and share parameters with other agents. Each agent's experience acquired through decentralized deployment is added to a common replay memory. During training, all the agent's experiences are optimized as mini-batches are chosen from the replay memory, which leads to faster convergence.

As defined in (14), our agents cannot be characterized as fully cooperative because the central element of the reward function, which has the same structure among the agents and is calculated with decentralized execution, is based on charging and jointly collected data [21]. Our setting is a simple cooperative where multi-agents learn the optimal cooperation policy.

Balancing the multiple objectives of data collection and charging can be much closer to being applied to the real world [22]. We found that simply adding individual components to the reward function was insufficient to achieve the two main objectives effectively. The advantage of transfer learning for agents is employed to learn how to balance multiple objectives effectively [13]. When there is no transfer learning in the first place, agents continue to collect data to receive positive rewards even if they run out of their batteries that are imposed to receive a sizeable negative reward in the near future. We use transfer learning to get the agent to learn their policy divided into two stages. In the first training, we set agents to not account for running out of their battery by removing the continuous battery decrease according to the mission time slot. In the next training, we continuously decreased their battery

according to the mission time slot when all other settings remained the same. Through transfer learning, agents could balance the two objectives by charging when needed.

The detailed procedure for the transfer learning part of our algorithm is presented in Algorithm 1. The reason why we test episodes in lines 10-11 of Algorithm 1 is that since various scenario parameters are included together in the training process of one control policy, only a few evaluations may reduce the reliability of the evaluation value due to extreme cases. Therefore, the moving average of the periodic multiple evaluations and the model weight separately when this updated value shows the best performance are accumulated. The weights are loaded during the second training phase or evaluation phase.

Algorithm 1 Training and Evaluation process based on transfer learning for the multiple objectives problem.

1: Initialize episode parameters.
2: Initialize experience replay memory $\mathbf{D}$.
3: Initialize the weights of original network θ and that of target network θ^{-}.
4: Fill replay memory with random action.
5: **Step-I: First training phase**
6: Set ϵ as 0.1.
7: Set decrement of the energy level of agent per step as 0.
8: **for** $epoch\ ep = 0, 1, \ldots$ **do**
9: Train episode.
10: **if** $ep\%5 = 0$ **then**
11: Test episode.
12: **end if**
13: **if** the best performance is achieved **then**
14: Save the model weights as θ'.
15: **end if**
16: **end for**
17: **Step-II: Second training phase**
18: Set ϵ to ranging from 0.5 to 0.01.
19: Set decrement of the energy level of agent per step as -1.
20: Load the model weights as θ'.
21: **Repeat** $8 - 16$.
22: **Step-III: Evaluation phase**
23: Load the model weights as θ'.
24: Hard update the target model weights as θ'.
25: **for** $epoch\ ep = 0, 1, \ldots,$ 1000 **do**
26: Test episode.
27: **end for**

V. SIMULATIONS

A. Settings

The UAV-oriented data collecting system on the sea surface was evaluated on a map that is discretized into 32×32 cells. At the start of each episode, scenario parameters that determine the scenario are sampled randomly within a predetermined range. The scenario parameters that can be specified are as follows: number of UAVs deployed, the initial amount of data to be collected from sensor nodes, flying time available for UAVs at the episode start, charging start capacity, charging complete capacity. The exact value ranges from which the randomized scenario parameters are depicted in Tab. II.

TABLE II: VALUE RANGES OF THE SCENARIO PARAMETERS.

Scenario Parameters	Min	Max
Number of deployed UAVs	1	3
Initial data volume to be collected	5.0	20.0
Maximum flying time	70	80
Charging start capacity	10%	50%
Charging complete capacity	60%	100%

TABLE III: EVENT REWARD STRUCTURE.

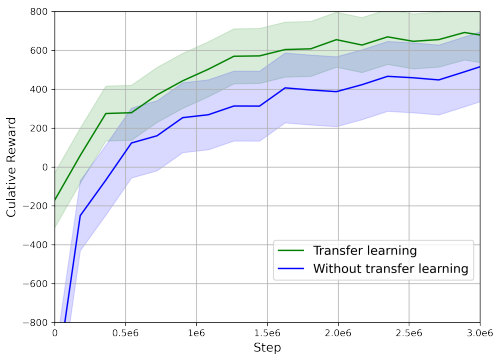
Symbol	Event	Reward
α	Data collection multiplier	+1.3
β	Agent collision with other agents or boundaries	-1.0
γ	Run out of energy	-300.0
δ	While charging	+0.05
ϵ	Communication quota is filled	+100.0
ζ	Standard movement	-0.1

The grid cell size c is 800 m, and the constant altitude h at which UAVs fly is 10 m. Each mission time slot contains four scheduled communication time slots, namely $m = 4$. Propagation parameters are chosen in line with [23] the rural macro scenario with $\alpha_{LoS} = 2.27$ and $\sigma^2_{LoS} = 2$.

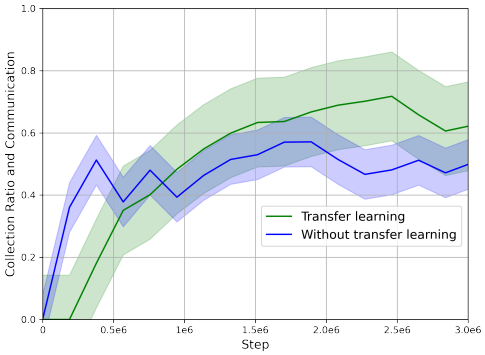
The network architecture of the proposed algorithm consists of convolution layers that accommodate $\mathbf{C}_{layer}$ and $\mathbf{D}_{layer}$ as inputs with ReLU activation that have a flatten layer that has 3 fully connected hidden layers containing 256 neurons respectively and a fully connected layer of which the size of the output is $\mathbf{A}$. The Q-values from the last fully connected layer are used as an input observation through the argument maximization policy while evaluating the agent and the softmax policy when training the agent.

The environment settings to implement the sensor nodes on the dynamic sea surface are as follows. Since it is a sustainable system using charging, one episode is set to end when the UAV energy level becomes empty or when a communication quota is fulfilled. We set the predefined communication quota $\mathbf{U}_C$ as 440. New data $\mathbf{D}_{k,new}$ allocated for each mission time slot to each sensor node follows a Poisson distribution, where $\lambda_p = 0.02$. The sensor node coordinates are 25 combinations of x-axis coordinates (3, 8, 13, 18, 23) and y-coordinates (6, 11, 16, 21, 26). The values of the exact parameters appeared in (14) are shown in Tab. III.

We use the following metrics to evaluate the agent's performance under different scenario instances. First, $successful\ communication$ collects whether all agents have achieved a communication quota at the end of an episode. Second, $collection\ ratio$ is the ratio of data collected until the end of the episode to the total data, which contains data given to sensor nodes at the beginning of the episode and the continuously increasing data. Lastly, $collection\ ratio\ and\ communication$ is the product of $successful\ communication$ and $collection\ ratio$ per episode.



(a) Cumulative reward per episode



(b) Data collection with successful communication per episode

Fig. 3: A comparison of the training process between transfer learning-based DRL and DRL systems without transfer learning. This curve includes the average and 99% confidence interval line shown through the exponential moving average. Note that the metrics are displayed with training steps equivalent to the number of epochs while training episode length is variable.

B. DRL path planning based on transfer learning

To demonstrate that transfer learning is essential to achieving higher training performance, we compare two training processes with and without transfer learning. When applying transfer learning, ϵ is tuned with e-greedy exploration starting at 0.5 and decaying to 0.01 in the second stage as shown in Algorithm 1. The reason for using decaying e-greedy exploration is that it was possible to meet the multiple objectives of charging and data collection experimentally.

Figure 3 shows the cumulative reward and the collection ratio with successful communication metric over the training step. The performance is high when transfer learning is used. In Fig. 3a, we can see that the transfer learning-based agent performs better in entire training steps. In Fig. 3b, it can be observed that a curve to which transfer learning is applied shows better performance in the end. We could observe that the curve to which transfer learning was not applied already converges from the 0.5M training step with low performance, whereas the curve to which transfer learning was applied outperformed in the 0.8M training step. This is because the transfer learning-based agent has already trained about collecting data, so the performance has risen steeper while training successful communication through charging. It is significant in that the performance when transfer learning is applied to both metrics is better.

TABLE IV: PERFORMANCE METRICS AVERAGED OVER 1000 RANDOM SCENARIO MONTE CARLO ITERATIONS.

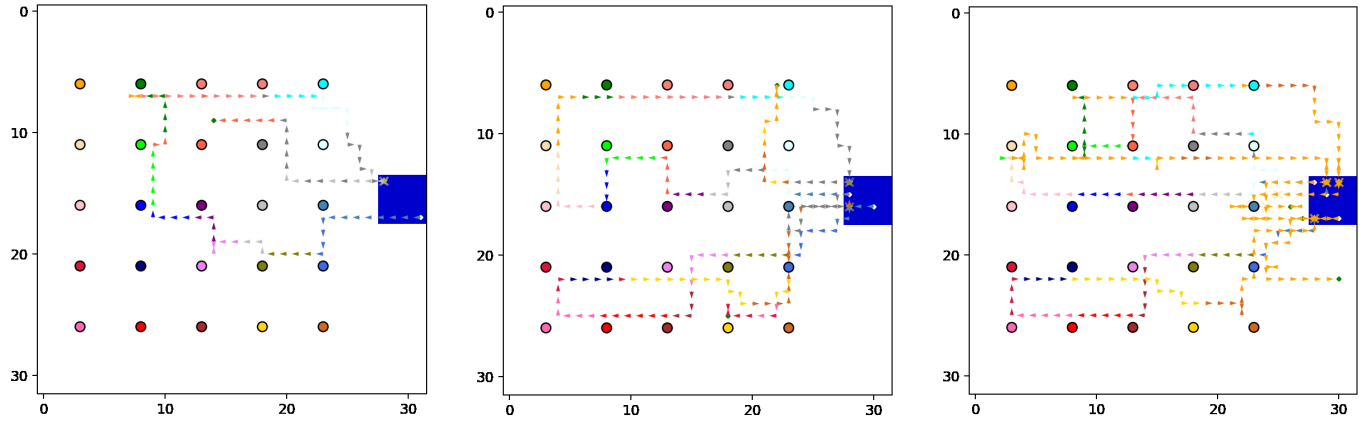
Metric	32×32 map
Successful Communication	90.1%
Collection Ratio	79.8%
Collection Ratio and Communication	71.9%

C. Visualization of simulation results

The scenario is presented based on the scenario parameter set in $Settings$ (see VI-A.). Performance in the scenario is evaluated using Monte Carlo simulations over the defined full range of scenario parameters. The overall average performance is shown in Tab. IV. Given that successful communication can be achieved through at least one charge, the agents show good successful communication performance. We believe that the collection ratio cannot be achieved at 100% because there are random scenario parameters that include flying time, number of deployed UAVs, and charging parameters and could be selected as an extreme combination, e.g. the number of deployed UAVs is one or starting charging when the remaining capacity is 10% as well as new data incremented for each sensor node until the end of the episode.

In Fig. 4, three example scenarios describe how the path planning was adapted as the number of deployed UAVs increased from 1 to 3. All other scenario parameters remain fixed at the values described in Fig. 4. The fixed scenario parameter values were set as the average value of the defined full range of the scenario parameters.

Fig. 4a shows a single agent trying to collect as much data as possible with one-time charging. The agent collects data while flying over the entire range of the sensor nodes. We can observe that a single agent has no choice but to ignore a few nodes containing the leftmost five nodes with x coordinate 3 and the bottommost 5 nodes with y coordinate 26 because sensor nodes are too widely deployed to cover all of these sensor nodes. Interesting points can be seen in Fig. 4b, where a second UAV is assigned. Contrary to Fig. 4a, it can be seen that two agents adapt the path planning to divide the entire sensor node into two sectors, which can be confirmed by the dramatic increase in the data collection ratio from 63.4% to 89.4% as well as the agents can fly to nodes far from the start and charging zone. It can be seen that successful communication has remained despite the difficulty of returning for charging in a multi-agent situation. In Fig. 4c, with the addition of the third UAV, the data collection ratio increased to 93.8%. However, successful communication decreased slightly due to a large number of agents.

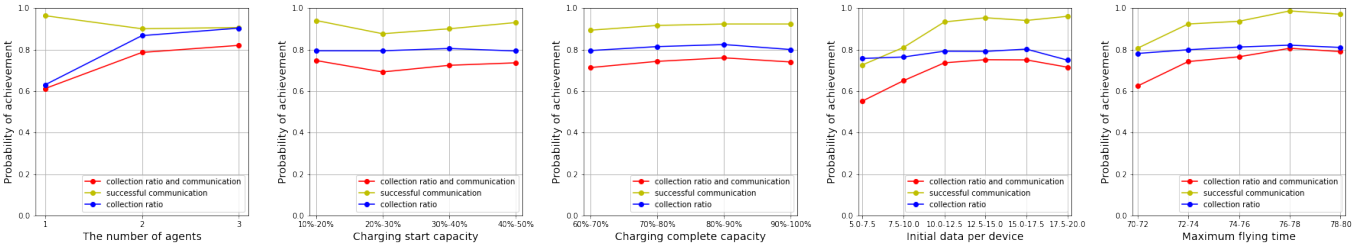


(a) I = 1, successful communication 99.6%, data collection ratio 63.4%

(b) I = 2, successful communication 99.6%, data collection ratio 89.4%

(c) I = 3, successful communication 99.0%, data collection ratio 93.8%

Fig. 4: Example trajectories for 32×32 map with charging start capacity = 30%, charging complete capacity = 80%, all sensor node with initial data volume = 12.5 and a maximum flying time = 80 steps.



(a) The number of agents. (b) Charging start. (c) Charging complete. (d) Initial data per device. (e) Maximum flying time.

Fig. 5: Effect on three metrics according to the scenario parameters. Each data point averages 300 Monte Carlo iterations for each scenario parameter space while maintaining the parameters to be investigated.

D. Effect of Scenario Parameters on Performance and Macro-level Benefits

What we want to emphasize is the effective reduction of training time. As demonstrated in [12] and [16], the global-local map approach has reduced the required network size enough for feasible learning. Also, training together with various scenario parameters helps find the sweet spot for each scenario. For example, in Fig. 4, it can be seen that Fig. 4c is the scenario closest to the sweet spot when deciding how many UAVs should be deployed with other scenario parameters given. The performance for specific parameters can be derived using the advantage of our approach to learning generalized path planning policy for various scenario parameters.

Fig. 5 shows the effect on three metrics for each specific scenario parameter. As demonstrated in the example trajectories are shown in Fig. 4, Fig. 5 shows that the performance is the best when the number of agents is three. As the number of agents increases further, we observed that the constantly generated data were collected well. In Fig. 5b, it can be seen that the successful communication is not lower than other charging start capacities even when the charging start capacity, which can be evaluated as adventurous in data collection, is 10%-20%. This is because the agents have learned enough within the defined range of scenario parameters. Similarly, in Figure 5c, it can be seen that the agents have learned well about all cases within the range of scenario parameters. As shown in Fig. 5d, it is beneficial to allocate a large amount of initial data per sensor node up to a range of [15.0, 17.5], but it is negative that the initial data is allocated over the range. Based on the reward structure (14), this is because there are fewer data to collect before [15.0, 17.5], and after the range, the collection ratio decreased because data per sensor node becomes abundant. Fig. 5e indicates that increasing the initial energy level for UAVs has a positive relationship with the collection ratio and communication.

However, we observed that when a large amount of communication quota was given in the multi-agent situation since the size of the charging area is small to 4×4 cells, the probability of successful communication achievement is reduced. In addition, the new data needs to be set larger in λ_p for the agents to collect data continuously. As a result, there is a complex relationship between UAV flying time, communication quota, and the amount of new data.

After fixing the parameter value showing the best performance in each scenario parameter range, it was confirmed that the collection ratio and communication was 0.93, the suc-

cessful communication was 0.99, and the collection ratio was 0.936 as a result of 300 Monte Carlo iterations. These values mean that the learning is carried out over the entire range of scenario parameters at once, so we can effectively check each parameter that performs optimally in the environment we defined and how much the best performance is at that time. In other words, sweet spots in the environment can be effectively found at the macro-level.

VI. CONCLUSIONS

We proposed a multi-agent deep reinforcement learning approach to build a sustainable system that UAVs can achieve with data harvesting at sensor nodes on the sea surface and charging when necessary. Through DDQN used with various scenario parameters and global-local map processing, it was confirmed that efficient learning was possible for agents in data collection, charging, and path planning constraints. The results show that the agent achieves multiple objectives balanced using a reward function design, transfer learning with a decaying e-greedy exploration technique. When a specific environment is defined, it can be seen from the results that it is possible to control scenario parameters at a macro level to show the best performance. In future work, we will specify the learning objective by collecting data considering the age of information collected from the sensor node. We will give each agent a unique network that makes an individual policy control operate in a decentralized deployment and decentralized training.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea [grant number 2021R1F1A1064059].

REFERENCES

[1] K. Liu, Z. Yang, M. Li, Z. Guo, Y. Guo, F. Hong, X. Yang, Y. He, Y. Feng, and Y. Liu, "Oceansense: Monitoring the sea with wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 14, no. 2, pp. 7–9, Sep. 2010.

[2] A. Alkandari, Y. Alabduljader, S. M. Moein *et al.*, "Water monitoring system using wireless sensor network (wsn): Case study of kuwait beaches," in *2012 Second International Conference on Digital Information Processing and Communications (ICDIPC)*. IEEE, Jul. 2012, pp. 10–15.

[3] M. Kishk, A. Bader, and M.-S. Alouini, "Aerial base station deployment in 6g cellular networks using tethered drones: The mobility and endurance tradeoff," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 103–111, Sep. 2020.

[4] C. A. Trasviña-Moreno, R. Blasco, Á. Marco, R. Casas, and A. Trasviña-Castro, "Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring," *Sensors*, vol. 17, no. 3, p. 460, Feb. 2017.

[5] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on uav communications for 5g and beyond," *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, Dec. 2019.

[6] S. Zhang, H. Zhang, B. Di, and L. Song, "Cellular uav-to-x communications: Design and optimization for multi-uav networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1346–1359, Jan. 2019.

[7] S. Shen, Y. Han, X. Wang, and Y. Wang, "Computation offloading with multiple agents in edge-computing–supported iot," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 1, pp. 1–27, Feb. 2019.

[8] G. Skorobogatov, C. Barrado, and E. Salamí, "Multiple uav systems: A survey," *Unmanned Systems*, vol. 8, no. 02, pp. 149–169, Apr. 2020.

[9] S. Fu, Y. Tang, Y. Wu, N. Zhang, H. Gu, C. Chen, and M. Liu, "Energy-efficient uav enabled data collection via wireless charging: A reinforcement learning approach," *IEEE Internet of Things Journal*, Jan. 2021.

[10] Z. Ullah, F. Al-Turjman, and L. Mostarda, "Cognition in uav-aided 5g and beyond communications: A survey," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 3, pp. 872–891, Jan. 2020.

[11] M.-A. Lahmeri, M. A. Kishk, and M.-S. Alouini, "Artificial intelligence for uav-enabled wireless networks: A survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1015–1040, Apr. 2021.

[12] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-uav path planning for wireless data harvesting with deep reinforcement learning," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1171–1187, May 2021.

[13] C. Punma, "Autonomous vehicle fleet coordination with deep reinforcement learning," Feb. 2018.

[14] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Uav path planning for wireless data harvesting: A deep reinforcement learning approach," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, Dec. 2020, pp. 1–6.

[15] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[16] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "Uav path planning using global and local map information with deep reinforcement learning," *arXiv preprint arXiv:2010.06917*, Oct. 2020.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[19] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, Mar. 2016.

[20] G. Chen, "A new framework for multi-agent reinforcement learning–centralized training and exploration with decentralized execution via policy distillation," *arXiv preprint arXiv:1910.09152*, Oct. 2019.

[21] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, Jun. 2021.

[22] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, Apr. 2019.

[23] 3GPP TR 38.901 version 14.0.0 Release 14, "Study on channel model for frequencies from 0.5 to 100GHz," *ETSI, Tech. Rep.*, May 2017.

VII. Biography Section

Mincheol Seong received the B.S. degree in Electrical Engineering from the Korea Military Academy, Korea, in 2019. He is currently a communication captain in the Republic of Korea Army. His research interests include military communications, WLAN systems, and machine learning.

Ohyun Jo received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2005, 2007, and 2011, respectively. He is currently an Associate Professor with the Department of Computer Science, Chungbuk National University. From April 2011 April to February 2016, he worked for Samsung Electronics in charge of research and development for future wireless communication systems, applications, and services. From March 2016 to July 2017, he was a Senior Researcher with the Electronics and Telecommunications Research Institute (ETRI), and from August 2017 to February 2018, he was an Assistant Professor with the Department of Electrical Engineering at Korea Military Academy. He has authored or coauthored more than 50 peer-reviewed papers, and holds more than 160 registered and filed patents. During his appointment at Samsung, He was a recipient of numerous recognitions including Gold Prize in Samsung Annual Award, the Most Creative Researcher of the Year Award, the Best Mentoring Award, Major R & D Achievement Award, and the Best Improvement of Organization Culture Award. His research interests include machine learning applications, millimeter wave communications, next generation WLAN/WPAN systems, 5G mobile communication systems, military communications, the Internet of Things, future wireless solutions/applications/services, and embedded communications ASIC design.

Kyungseop Shin received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), in 2009, 2011, and 2015, respectively. He is currently an Assistant Professor with the Department of Computer Science, Sangmyung University. From March 2015 to August 2017, he was a Senior Researcher with Korea Telecommunication in charge of research and development for 5th generation communication systems and services. From September 2017 to March 2020, he was an Assistant Professor with the School of Computer Science, Semyung University. His current research interests include next generation communication systems, WLAN systems, the Internet of Things, and machine learning based network capacity and lifetime maximization.

Declaration of interests

☐The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☒The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

MinCheol Seong reports financial support was provided by The National Research Foundation of Korea.

Mincheol Seong

September 6, 2022

Dear the editor-in-chief:

Please accept this letter as an indication of my sincere interest in the ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE. I received the B.S. degree in Electrical Engineering from the Korea Military Academy, Korea, in 2019. I am currently a communication captain in the Republic of Korea Army. My research interests include military communications, WLAN systems, and machine learning.

After graduating from electronic engineering, I learned and studied AI and networks under the guidance of professors Ohyun Jo and Kyungseop Shin. In particular, I was interested in Reinforcement Learning and UAV-aided networks, so I intensively studied them in related fields. I kept records of my research on GitHub. See https://github.com/mincheolseong. In the '5SO_makeathon' repository on GitHub, I used Reinforcement Learning at an AI contest hosted by the Republic of Korea Army. I won first place with the development of an optimal trajectory analysis algorithm, and related codes are also attached. There is also a record of giving a special lecture on AI at the Army Information and Communication School in the same repository. In addition, I have written several codes with regard to multi-agent, DQN, and communication themes.

Although I have only completed the bachelor's course, I have studied Reinforcement Learning and UAV-aided network for about two to three years. I have achieved results such as winning the competition. Writing the thesis is also for admission to graduate school admission in the future, and I do have a strong desire for a higher level of study.

Thank you for your time and consideration. I can be reached via email (smc5741@gmail.com).

Sincerely,

Mincheol Seong