Meta-Learning Deep Reinforcement Learning for Fresh Data Collection in UAV-Assisted Wireless Sensor Networks

Xiong Xiao *, Mengjie Yi *, Xijun Wang †, Juan Liu ‡, Yan Zhang §, Ronghui Hou *
*School of Cyber Engineering, Xidian University, Xi'an 710071, China
†School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, 510006, China
‡School of Electrical Engineering and Computer Science, Ningbo University, Zhejiang 315211, China
§State Key Lab of Integrated Service Networks,
Information Science Institute, Xidian University, Xi'an, Shaanxi, 710071, China
Email: xiongxiao@stu.xidian.edu.cn, mjyi@stu.xidian.edu.cn, wangxijun@mail.sysu.edu.cn, eeliujuan@gmail.com,
yanzhang@xidian.edu.cn, rhhou@xidian.edu.cn

***Abstract*—Using unmanned aerial vehicles (UAVs) to collect data from large-scale sensor nodes (SNs) in wireless sensor networks (WSNs) is a practical approach. However, since SNs may change locations due to disasters (e.g., landslides or earthquakes), quickly responding to these dynamic network scenarios presents a significant challenge. In this paper, we address the problem of timely data collection from SNs in WSNs by optimizing the UAV's trajectory, aiming to minimize the weighted sum of the average age of information (AoI) and the maximum AoI. Specifically, SNs are grouped into square-shaped clusters, with the UAV flying over the square center points (SCPs) to collect data from the SNs in each cluster. When the positions of the SNs change, the UAV can rapidly adjust its trajectory to efficiently collect fresh data. We formulate this problem as a Markov decision process (MDP) with non-uniform time steps and propose a meta-learning deep reinforcement learning (MLDRL) method to solve it. Simulation results demonstrate that the MLDRL-based algorithm not only achieves the fastest convergence when the scenario changes but also outperforms baseline methods in overall performance.**

***Index Terms*—Unmanned aerial vehicles, age of information, wireless sensor networks, meta learning, deep reinforcement learning.**

I. Introduction

Wireless sensor networks (WSNs) are widely utilized in fields such as environmental monitoring, event detection, and target tracking [1]. Typically, WSNs comprise numerous sensor nodes (SNs), each powered by a small-capacity battery and equipped with limited computational and networking capabilities. UAVs, with fully controllable mobility, can fly close to SNs to collect data, thereby reducing the energy consumption of the SNs and enhancing the data transfer rate [2]. In critical applications such as disaster monitoring and rescue, maintaining the freshness of information is paramount, which is why the age of information (AoI) is employed as a key metric for evaluating data timeliness [3].

Recent studies have investigated how to preserve data freshness in UAV-assisted data collection within WSNs. In [4], a deep reinforcement learning (DRL) algorithm was implemented to optimize UAV trajectory, SNs scheduling, and charging decisions to minimize AoI. However, this work considers only a small number of SNs, making it unsuitable for scenarios involving large-scale SNs. A method to improve efficiency in large-scale SNs involves clustering SNs, allowing a UAV to collect data by visiting just the cluster heads instead of each SN [5]–[7]. However, studies [5]–[7] assumed fixed SN positions; they do not account for potential shifts in SN positions due to disasters, which can lead to outdated data collection strategies and compromise data freshness. In such environments where the possibility of change exists, data collection policies must be swiftly adjusted to maintain data freshness. Traditional methods, however, may take significant time to recalibrate, struggling to adapt quickly to environmental changes.

Driven by these opportunities and challenges, we delve into disaster monitoring in WSNs using a meta learning DRL (MLDRL) approach. This approach focuses on learning the trajectory of the UAV that not only ensures the collection of fresh data but also enable rapid adaptation to environmental changes during a disaster. Specifically, we divide the entire area of interest into multiple equally sized square sub-areas (SSAs) where UAVs can fully cover the communication. The UAV flies over the SSA center points (SCPs) and collects data packets from SNs in each SCP during the designated time. Our goal is to minimize the weighted sum of the average and maximum AoI of all SNs by optimizing the UAV's trajectory. In the event of a disaster, it's possible for some SNs to shift locations, requiring the UAV to quickly identify a new efficient trajectory to maintain the freshness of collected data. To address this, we model the problem as a

This work was supported in part by the National Natural Science Foundation of China under Grants 42127802 and 62271513, the National Science Foundation under Grant U23B2024, the Zhejiang Provincial Natural Science Foundation of China under Grant LY24F010002, and the Ningbo Natural Science Foundation under Grant 2023J123.

ISBN 978-3-903176-65-2 © 2024 IFIP

Markov decision process (MDP) with nonuniform time steps. Leveraging the rapid adaptability of meta-learning [8], we introduce a MLDRL-based data collection problem. Extensive simulations demonstrate that our algorithm not only surpasses baseline algorithms in convergence speed but is also more effective than the baseline algorithms when the locations of some SNs are changed.

This paper is organized as follows: Section II presents the system model and problem formulation. Section III proposes the MLDRL-based UAV data collection algorithm. Section IV discusses the simulation results. Section V concludes the paper.

II. System Model and Problem Formulation

A. Scenario Description

As shown in Fig. 1, we study a UAV-asssisted WSN, where N SNs are deployed over a square ground area with length L. Each SN i is denoted as n_i, where $1 \leq i \leq N$, and the set of all SNs is represented as $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$. The position of SN n_i is denoted as $\boldsymbol{L}_i^n = (x_i, y_i), \forall i \in \mathcal{N}$. When natural disasters occur, such as landslides or earthquakes, the positions of some SNs may shift. A UAV takes off from a hangar to collect data from SNs. The UAV has a communicable radius of r on the ground, which means that it can collect data from SNs on the ground within a circular area of radius r centred on its projection. We discretize this area into J SSAs, each of which has a side length of l, where $J = \lceil L/l \rceil^2$ and $l = 2\sqrt{2}r$. This ensures that the communication radius r of the UAV can cover the entire small square. We assume that the flight altitude of the UAV is fixed and denoted as h. The UAV can hover over SCPs to collect data. The set of all SCPs is denoted $\mathcal{C} = \{c_1, c_2, \ldots, c_J\}$, where c_j denotes the j-th SCP. Note that among the J SCPs, there are some SCPs that may not have SNs distributed. The position of the center of the j-th small square denoted as $\boldsymbol{L}_j^c = (x_j, y_j)$, where $x_{\min} \leq x_j \leq x_{\max}$ and $y_{\min} \leq y_j \leq y_{\max}$, $x_{\min}$/$y_{\min}$ and $x_{\max}$/$y_{\max}$ denote minimum and maximum values in the horizontal/vertical direction, respectively. We refer to SCPs with SNs distribution as effective SCPs (ECPs) and SCPs without SNs distribution as invalid SCPs (ICPs). Thus, the set of ECPs and ICPs are denoted as $\overline{\mathcal{C}} = \{\bar{c_1}, \bar{c_2}, \ldots, \bar{c_G}\}$ and $\hat{C} = \{\hat{c_1}, \hat{c_2}, \ldots, \hat{c_F}\}$, where G and F represent the number of the ECPs and ICPs, respectively, satisfy $G + F = J, \overline{\mathcal{C}} \cup \hat{C} = \mathcal{C}$. The position of the center of the g-th ECP denoted as $\boldsymbol{L}_g^c = (x_g, y_g)$. Without loss of generality, the set of SNs in SCP j is denoted as $\mathcal{N}_j = \{n_{1,j}, n_{2,j}, \ldots, n_{K_j,j}\}$, where $n_{k,j}$ represents the k-th SN in SCP j, $K_j = |\mathcal{N}_j|$, and $\sum_{j=1}^{J} K_j = N$.

The duration of the data collection task is T seconds. At the beginning of the task, the UAV takes off from the hangar $c_1 \in \mathcal{C}$, flies along SCPs to collect data packets from SNs. The data collection task of the UAV involves three phases: decision-making, flight and data collection. In the decision-making phase, the UAV decides on its direction of flight, which is one of four: up, down, left, and right. In the flight phase, the UAV flies to its neighbouring SCP with a fixed speed v in the direction chosen in the decision-making phase. In the data collection phase, if there are SNs in the square of the SCP reached by the UAV, then the UAV hovers over this SCP to collect data on all SNs in this small square, then restart to decision-making phase. If there is no SN, then the UAV will restart the decision-making phase. Let $\boldsymbol{M}$ denote the sequence of SCPs visited by the UAV, and $\boldsymbol{M} = (c^1, c^2, \ldots, c^m)$, where $c^z, 1 \leq z \leq m$, represents the z-th visited SCP and m denotes the number of SCPs visited by the UAV in the during T seconds. Note that the value of m is related to the trajectory of the UAV and that a SCP can be visited multiple times in time T. Because the number of SNs in each SCP may be different, the time consumed by the UAV in the data collection phase on each SCP may be different. Therefore, the time consumed by the UAV to travel from one SCP to another of its neighbouring SCPs may be different. Let $\boldsymbol{L}(t_z) = (x(t_z), y(t_z))$ denote the projection of the UAV's position on the ground at time t_z, the trajectory of the UAV is defined as $\boldsymbol{U} = (\boldsymbol{L}(t_1), \boldsymbol{L}(t_2), \ldots, \boldsymbol{L}(t_m))$, where t_z denotes the time when the UAV completes data collection in the z-th SCP in its visiting sequence $\boldsymbol{M}$ and $\boldsymbol{L}(t_1) = \boldsymbol{L}_1^c$.

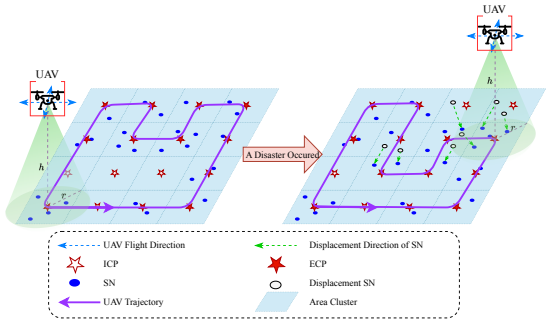

Figure 1. A scenario of a UAV collecting data in a WSN prone to natural disasters.

B. Channel Description

Wireless communication between the UAV and SNs needs to consider both line-of-sight (LoS) and non-line-of-sight (NLoS) channel conditions. When the UAV is hovering at SCP c_j, assuming n_i is within c_j and is also $n_{k,j}$, the Euclidean distance between the UAV and $n_{k,j}$ can be expressed as $d_{k,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + h^2}$. The LoS probability between the UAV and the SN $n_{k,j}$ depends on the communication distance, elevation angle, and channel environment, which is expressed as [7]

$$\mathbb{P}_{k,j}^{L}(d_{k,j}) = \frac{1}{1 + a\exp\left(-b\left[\frac{180}{\pi}\arcsin\left(\frac{h}{d_{k,j}} - a\right)\right]\right)}, \quad (1)$$

where a and b are constants determined by the environment. The NLoS probability between the UAV and the SN $n_{k,j}$ is $\mathbb{P}_{k,j}^{NL}(d_{k,j}) = 1 - \mathbb{P}_{k,j}^{L}(d_{k,j})$. Therefore, the path loss of the air-to-ground wireless communication channel can be calculated as

$$\mathfrak{L}\left(d_{k,j}\right)=\left(\frac{4\pi f_c d_{k,j}}{v_c}\right)^2 \beta_w, w \in \{\text{LoS}, \text{NLoS}\}, \quad (2)$$

where f_c is the carrier frequency, v_c is the speed of light, and β_{LoS} and β_{NLoS} represent the excessive path losses under LoS and NLoS conditions, respectively, with $1 < \beta_{\text{LoS}} < \beta_{\text{NLoS}}$.

The information collection process at each SN is event-driven. That is, the UAV establishes communication links with the SNs within the SCP individually only after reaching the SCP. Let P represent the transmission power of SNs, and the upload data rate of the SN $n_{k,j}$ can be expressed as

$$R_{k,j}=B\log_2(1+\frac{\delta P}{\mathfrak{L}\left(d_{k,j}\right)\cdot\sigma^2}), \quad (3)$$

where δ is the channel gain at one meter in space, B and σ^2 respectively represent the system bandwidth and the noise power. Let $\mathscr{L}$ denote the packet size of SN $n_{k,j}$, and the upload time of SN $n_{k,j}$ can be expressed as $\xi_{k,j}=\mathbb{E}\left[\mathscr{L}/R_{k,j}\right]$, where $\mathbb{E}[\cdot]$ is the expectation of the path loss.

We assume that the SNs in the SCP can sequentially upload their sensed data to the UAV via a time division multiple access (TDMA) scheme. Therefore, the time consumed by the UAV during the data collection phase hovering at SCP c_j can be expressed as

$$\varphi_j=\begin{cases}\sum_{k=1}^{K_j}\xi_{k,j}, & \text{if } c_j\in\overline{\mathcal{C}},\\ 0, & \text{otherwise.}\end{cases}. \quad (4)$$

C. Age of Information

Let $\Lambda_i(t_z)$ denote the AoI of the SN n_i at time t_z. If n_i is within c_j and is also $n_{k,j}$, the AoI of $n_{k,j}$ at time t_z is given by

$$\Lambda_{k,j}(t_z)=(t_z-t_z^{k,j})^+, \quad (5)$$

where $t_z^{k,j}$ represents the time at which $n_{k,j}$ samples and generates information as the UAV arrives at the z-th SCP, with $t_z^{k,j}\leqslant t_z$, and $(a)^+=\max\{0,a\}$. Therefore, the average AoI $\overline{\Lambda}(t_z)$ and maximum AoI $\Lambda_{\max}(t_z)$ of all SNs at time t_z can be expressed as

$$\overline{\Lambda}(t_z)=(1/N)\sum_{j=1}^{J}\sum_{k=1}^{K_j}\Lambda_{k,j}(t_z), \quad (6)$$

$$\Lambda_{\max}(t_z)=\max\left\{\Lambda_{k,j}(t_z), 1\leq j\leq J, 1\leq k\leq K_j\right\}. \quad (7)$$

If the UAV is in SCP j at time t_z, i.e., $\boldsymbol{L}(t_z)=\boldsymbol{L}_j^c$, the sensed data of all SNs in SCP j have been uploaded to the UAV. Then, the k-th SN in SCP j updates its AoI to $\sum_{k'=k}^{K_j}\xi_{k',j}$. Otherwise, the AoI of the k-th SN in SCP j is increased by $\phi+\varsigma+\varphi_{j'}$, where ϕ is the time required for decision-making at time t_{z-1}, ς is the time required for the flight, $\varsigma=\frac{l}{v}$, and j' denotes the SCP that the UAV hovering at time t_z, $j'\neq j$. Therefore, the update of the AoI at t_z can be expressed as

$$\Lambda_{k,j}(t_z)=\begin{cases}\sum_{k'=k}^{K_j}\xi_{k',j}, & \text{if } \boldsymbol{L}(t_z)=c_j,\\ \Lambda_{k,j}(t_{z-1})+\phi+\varsigma+\varphi_{j'}, & \text{otherwise.}\end{cases} \quad (8)$$

D. Problem Formulation

We aim to identify the optimal trajectory of the UAV to minimize the weighted sum of the average and maximum AoI across all SNs. The optimization problem is formulated as

$$\mathscr{P}_1: \min_{\boldsymbol{U}}\frac{1}{m}\sum_{z=1}^{m}\left(\varepsilon_1\overline{\Lambda}(t_z)+\varepsilon_2\Lambda_{\max}(t_z)\right), \quad (9)$$

$$s.t. \boldsymbol{L}(t_1)=\boldsymbol{L}_1^c, \quad (10)$$

$$t_m\leq T, \quad (11)$$

$$x_{\min}\leq x(t_z)\leq x_{\max}, \quad (12)$$

$$y_{\min}\leq y(t_z)\leq y_{\max}, \quad (13)$$

where ε_1 and ε_2 are the weights of the average AoI and the maximum AoI of all SNs, respectively. (10) ensures that the UAV ensures that the UAV departs from its hangar. (11) guarantees that the time for the UAV to finish data collection in m-th SCP cannot exceed the total time T. (12) and (13) guarantee that the UAV remains within the region of interest throughout its flight. To solve this continuous, multi-stage decision problem, we propose a DRL-based algorithm. However, when a disaster occurs and the environment changes, traditional reinforcement learning methods need to start training again, requiring high time and computational costs to converge. To address this challenge and improve the convergence speed of deep reinforcement learning algorithms in changing environments, we propose a meta-learning DRL-based algorithm.

III. Meta-Learning DRL-based Approach

In this section, we cast the UAV-assisted data collection problem into a Markov decision process (MDP) with nonuniform time steps firstly and then propose a Meta-Learning DRL-based (MLDRL) algorithm to solve it.

A. MDP Formulation

An MDP can be described by a tuple $(\mathcal{S},\mathcal{A},r,\mathcal{P})$, where $\mathcal{S},\mathcal{A},r,\mathcal{P}$ represent the state space, action space, reward function, and state transition function, respectively. At time t_z, the UAV (agent) observes the environment's state $s(t_z)\in\mathcal{S}$ and performs an action $a(t_z)\in\mathcal{A}$. Then, the current state transitions to the next state at time t_{z+1} with probability $p\left(s(t_{z+1})|s(t_z),a(t_z)\right)$, and the agent receives a reward $r(t_z)$. We will now detail the states, actions, reward function, and state transitions.

1) State: The state at time t_z is defined as $s(t_z)=(\boldsymbol{\Lambda}(t_z),\boldsymbol{\Delta}(t_z),\Xi(t_z))$, where

- $\boldsymbol{\Lambda}(t_z)$ is the AoI of all SNs time t_z, expressed as $\boldsymbol{\Lambda}(t_z)=\{\Lambda_{k,g}(t_z)\mid 1\leq k\leq K_g, 1\leq g\leq G\}$.
- $\boldsymbol{\Delta}(t_z)$ represents the difference in coordinates between the UAV's position $\boldsymbol{L}(t_z)$ and all SCPs, expressed as $\boldsymbol{\Delta}(t_z)=(\boldsymbol{L}(t_z)-\boldsymbol{L}_1^c,\boldsymbol{L}(t_z)-\boldsymbol{L}_2^c,\ldots,\boldsymbol{L}(t_z)-\boldsymbol{L}_J^c)$.

- $\Xi(t_z)$ is the remaining time for the UAV, expressed as $\Xi(t_z) = T - t_z$.

2) Action: The action of the UAV in the decision-making phase at time t_z is defined as $a(t_z) \in \mathcal{A} \triangleq \{\text{up}, \text{down}, \text{left}, \text{right}\}$.

3) Reward Function: The reward function should encourage the UAV to minimize the weighted sum of the average AoI and maximum AoI of all SNs under the constraints given in (10)-(13). Once the constraints are violated, the UAV will be penalized. Let $\rho(t_z)$ denote whether the UAV's action at the time t_z will cause the UAV's position to be outside the region of interest. In particular, if the UAV's action at time t_z will cause the UAV's position to be out of the region of interest, $\rho(t_z) = 1$; otherwise, $\rho(t_z) = 0$. The reward function is then expressed as

$$r(t_z) = \begin{cases} -e, & \text{if } \rho(t_z) = 1, \\ \frac{\varepsilon_1}{1+\overline{\Lambda}(t_z)} + \frac{\varepsilon_2}{1+\Lambda_{\max}(t_z)}, & \text{otherwise}, \end{cases} \tag{14}$$

Where e is a constant, 1 is added to both denominators in the reward function to prevent calculation errors when the denominators are zero in the initial state.

4) State Transition: The update of AoI is described in (8). The update of $\boldsymbol{\Delta}(t_z)$ is determined by the position of the UAV. The dynamics of the position of the UAV can be described as

$$\boldsymbol{L}(t_{z+1}) = \begin{cases} \boldsymbol{L}(t_z), & \text{if } \rho(t_z) = 1, \\ \boldsymbol{L}(t_z) + (0, l), & \text{if } \rho(t_z) = 0 \text{ and } a(t_z) = \text{up}, \\ \boldsymbol{L}(t_z) + (0, -l), & \text{if } \rho(t_z) = 0 \text{ and } a(t_z) = \text{down}, \\ \boldsymbol{L}(t_z) + (-l, 0), & \text{if } \rho(t_z) = 0 \text{ and } a(t_z) = \text{left}, \\ \boldsymbol{L}(t_z) + (l, 0), & \text{if } \rho(t_z) = 0 \text{ and } a(t_z) = \text{right}. \end{cases} \tag{15}$$

The update of the remaining time $\Xi(t_z)$ is determined by the time t_z. We assume that when the action of the UAV at time t_z will take it out of the range of interest, the UAV's position will remain unchanged, but the time will experience $\phi + \varsigma$. Therefore, the update of the time t_z is expressed as

$$t_{z+1} = \begin{cases} t_z + \phi + \varsigma, & \text{if } \rho(z) = 1, \\ t_z + \phi + \varsigma + \varphi_{j^-}, & \text{otherwise}, \end{cases} \tag{16}$$

where φ_{j^-} represents the time spent by the UAV hovering over the SCP c_{j^-} in the data collection phase at time t_{z+1}.

The objective of an MDP is to discover an optimal policy that, starting from the initial state $s(0)$, maximizes the discounted rewards over the entire episode. Specifically, the optimal policy of the MDP is defined as

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{z=1}^{m} r(t_z)|s(0)\right], \tag{17}$$

where the expectation is calculated based on the distribution across state and action sequences, influenced by policy π and the associated transition probabilities.

B. Meta-Learning DRL Approach

DRL can deal with the aforementioned MDP. However, traditional DRL-based algorithms are typically designed for scenarios where locations of SNs remain constant. In situations where some SNs shift due to disasters, strategies derived from DRL method might not suit the altered environment. Meta learning allows the DRL method to quickly adapt to new environment, utilizing previous empirical knowledge and reducing the reliance on extensive training data [8]. Therefore, we propose a meta-learning DRL (MLDRL) approach to solve this UAV-assisted data collection problem.

In this work, the environment prior to a disaster represents an initial task. When a disaster occurs, the locations of some SNs are changed, resulting in a task that differs from the initial one. In the MLDRL, the agent aims to acquire the meta-parameters $\boldsymbol{\theta}$ by utilizing multiple tasks from a given task distribution $p(\mathcal{T})$. Task $\mathcal{T}_i \sim p(\mathcal{T})$ can be described as an MDP model defined by $(\mathcal{S}_i, \mathcal{A}_i, r_i, \mathcal{P}_i)$, where $\mathcal{S}_i$, $\mathcal{A}_i$, r_i, and $\mathcal{P}_i$ represent the state space, action space, reward function, and state transition function of $\mathcal{T}_i$, respectively. For each task $\mathcal{T}_i$, the meta-parameters $\boldsymbol{\theta}$ are updated to $\boldsymbol{\theta}_i$ by utilizing the training trajectories D_i^{tr}. D_i^{tr} denotes the set of W long trajectories sampled over the meta-parameters $\boldsymbol{\theta}$ for task $\mathcal{T}_i$, where $D_i^{tr} = \{D_{i,w}^{tr} | 1 \le w \le W\}$, $D_{i,w}^{tr} = \{s_{i,w}(t_z), a_{i,w}(t_z), r_{i,w}(t_z) \mid 1 \le z \le m_{i,w}\}$, and $m_{i,w}$ represents the number of SCPs visited by the UAV in the w-th trajectory in $\mathcal{T}_i$. The loss function of the task-learner for task $\mathcal{T}_i$ is

$$\mathcal{L}_i^1(\boldsymbol{\theta}, D_i^{tr}) = \frac{1}{W\overline{m}_i} \sum_{w=1}^{W} \sum_{z=1}^{m_{i,w}} \mathbb{E}\Bigg[\log \pi(a_{i,w}(t_z)|s_{i,w}(t_z); \boldsymbol{\theta}) \times \hat{A}(s_{i,w}(t_z))\Bigg], \tag{18}$$

where $\overline{m}_i = \max\{m_{i,w} \mid 1 \le w \le W\}$, $\hat{A}(\boldsymbol{s}_{i,w}(t))$ is the generalized advantage estimation of $s_{i,w}(t_z)$ [9]. Then, the updated $\boldsymbol{\theta}_i$ is obtained by

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta} - \alpha_1 \nabla_{\boldsymbol{\theta}} \mathcal{L}_i^1(\boldsymbol{\theta}, D_i^{tr}), \tag{19}$$

where α_1 is the learning rate of the task learner.

The updated parameters $\boldsymbol{\theta}_i$ are utilized to collect the validation trajectories D_i^{vd} to update the meta-parameters $\boldsymbol{\theta}$. The loss function of the meta-learner for task $\mathcal{T}_i$ is [10]

$$\begin{aligned} \mathcal{L}_i^2(\boldsymbol{\theta}_i, D_i^{vd}) = & \frac{1}{W\overline{m}_i} \sum_{w=1}^{W} \sum_{z=1}^{m_{i,w}} \mathbb{E}\Bigg[\frac{\pi(a(t_z)|s(t_z); \boldsymbol{\theta}_i)}{\pi(a(t_z)|s(t_z); \boldsymbol{\theta}_i^{old})} \times, \\ & \hat{A}(s(t_z))\Bigg], \\ s.t. \quad & \mathbb{E}\left[\text{KL}[\pi(\cdot|s(t_z); \boldsymbol{\theta}_i^{old}), \pi(\cdot|s(t_z); \boldsymbol{\theta}_i)]\right] \le \epsilon, \end{aligned} \tag{20}$$

where $\boldsymbol{\theta}_i^{old}$ is the parameters before the update of $\boldsymbol{\theta}_i$, $\text{KL}[\pi_1, \pi_2]$ represents the KL divergence between two policies π_1 and π_2, ϵ denotes the maximum KL divergence constraint.

Then, the meta-learner summarizes the loss functions of each task to update the meta-policy, which is expressed as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_2 \nabla_{\boldsymbol{\theta}} \left(\sum_{i=1}^{I} \mathcal{L}_i^2(\boldsymbol{\theta}_i, D_i^{vd}) \right), \tag{21}$$

where α_2 is the meta learning rate and I is the number of tasks.

The process of the MLDRL-based algorithm is detailed in Algorithm 1. First, we initialize the task distribution $p(\mathcal{T})$, the number of meta-iterations M, the number of tasks per meta-iteration I, the number of trajectories sampled per task W, and the meta-parameters $\boldsymbol{\theta}$. During the meta-training phase, the task-parameters $\boldsymbol{\theta}_i$ and the meta-parameters $\boldsymbol{\theta}$ are updated alternately. Specifically, we sample I tasks from the task distribution (Line 2). We generate W training trajectories D_i^{tr} using $\boldsymbol{\theta}$ on each task $\mathcal{T}_i$ and calculate the loss function based on (18). The task parameters $\boldsymbol{\theta}_i$ for task $\mathcal{T}_i$ are updated by performing one or more gradient updates according to (19) (Lines 3~7). Using the updated task parameters $\boldsymbol{\theta}_i$, the validation trajectories D_i^{vd} is generated, and the meta-learner's loss function is computed based on (20) (Lines 8~11). The meta-parameters $\boldsymbol{\theta}$ are updated by performing one or more gradient updates according to (21) (Line 12).

Algorithm 1 MLDRL-based UAV-assisted timely data collection algorithm.

Require: Initialize $p(\mathcal{T})$, M, I, W, and $\boldsymbol{\theta}$;
1: **for** $n_{meat} = 1 : N_{meta}$ **do**
2: Sample I tasks, $\mathcal{T}_i \sim p(\mathcal{T}), i \in I$;
3: **for** each task $\mathcal{T}_i$ **do**
4: Sample W training trajectories D_i^{tr} using $\boldsymbol{\theta}$ in $\mathcal{T}_i$;
5: Compute the loss function $\mathcal{L}_i^1$ using D_i^{tr} according to (18);
6: Update parameters $\boldsymbol{\theta}_i$ using according to (19);
7: **end for**
8: **for** each task $\mathcal{T}_i$ **do**
9: Sample a new validation trajectories D_i^{vd} using $\boldsymbol{\theta}_i$ in $\mathcal{T}_i$;
10: Compute the loss function of the meta-learner $\mathcal{L}_i^2$ using D_i^{vd} according to (20);
11: **end for**
12: Update $\boldsymbol{\theta}$ according to (21);
13: **end for**

IV. Simulation Results

In this section, we conduct extensive simulations to evaluate the performance of the MLDRL-based UAV-assisted data collection algorithm in WSNs. We divide a square region of interest with an area of 1000m $\times$ 1000m into 20×20 grids of equal-sized cells, each of which a length of 50 meters. The center of the cell in the lower left corner is the hangar of the UAV. All SNs are randomly distributed in the geographical area of interest, and all SNs have the same importance weight.

Table I
SYSTEM PARAMETERS AND HYPERPARAMETERS OF THE MLDRL

Parameters	Value
L, h, r, v, T	1000 m, 62 m, 36 m, 20 m/s, 1800 s
$a, b, c, \beta_{\text{LoS}}, \beta_{\text{NLoS}}$	9.61, 0.16, 3×10^8 m/s, 0 dB, 3 dB
$P, B, \sigma^2, \beta, \mathscr{L}$	0.1 W, 10 MHz, -110 dBm, -60 dB, 10^6 bit
$\varepsilon_1, \varepsilon_2, e$	10, 10, 2
$M, I, W, \alpha_1, \alpha_2, \epsilon$	100, 20, 20, 0.99, 0.95,0.01
Optimizer	Adam
Activation function	Tanh

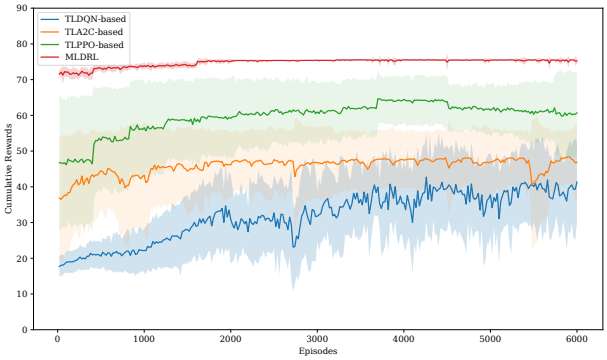

Figure 2. Evaluation of the convergence performance of the proposed MLDRL-based algorithm and other DRL-based algorithms on new tasks. ($N = 120$).

Unless otherwise specified, the simulation parameters are as shown in Table I.

The proposed algorithm's performance is compared against four different baseline algorithms. The first is Deep Q-Networks (DQN) alogorithm [11]. The second is Advantage Actor Critic (A2C) alogorithm [12]. The third is Proximal Policy Optimization (PPO) alogorithm [9]. To enhance the performance of traditional DRL algorithms on new tasks, we integrate transfer learning with above DRL algorithms, resulting in TLDQN, TLA2C, and TLPPO, respectively. These DRL algorithms are trained in the initial environment, and then the learned policies are transferred to the new task [13]. The forth is AoI-based algorithm, In this algorithm, the UAV always flies towards the SCP where the SN with maximum AoI, and the UAV can collect data on the SNs of the SCPs it passes by on its way to the SCP with the maximum AoI.

The proposed algorithm's neural networks are implemented based on PyTorch, and both the proposed MLDRL-based algorithm and baseline algorithms, implemented on the platform in [14], are trained and evaluated in parallel with 20 workers. For both MLDRL and DRL-based algorithms, the policy network is a deep neural network with two hidden layers, each of which consists of 512 neurones.

Fig. 2 presents the convergence process of the proposed algorithm on a new task after training, where the horizontal axis shows the number of episodes and the vertical axis denotes the cumulative reward in an episode. The three baseline algorithms are initially trained in the original environment, and then their learned policies are transferred to the new task. It's important

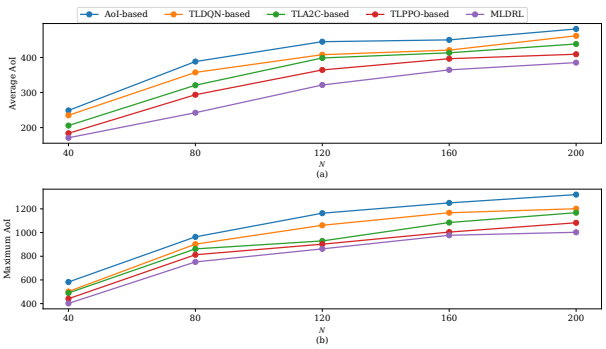

Figure 3. Comparison of the performance of the proposed MLDRL-based algorithm and four baseline algorithms under different numbers of SNs N.

to note that all algorithms retain the capability to learn when assessed on the new task. In Fig. 2, the solid line indicates the average cumulative reward, while the shaded area reflects the variance. Higher initial positions of the solid line suggest superior initial adaptation to the new task, whereas lower positions imply poorer adaptation. Broader shaded areas represent more significant fluctuations in the algorithm's response to the new task, indicating reduced stability. From Fig. 2, it is evident that our proposed MLDRL algorithm achieves faster convergence, higher average cumulative rewards, and greater stability in the new task compared to other baseline learning algorithms. This superiority stems from MLDRL's ability to rapidly adjust to new tasks by leveraging more generalized knowledge acquired from different tasks during training.

Fig. 3 illustrates how the performance of various algorithms varies with different numbers of SNs N. Specifically, Fig. 3(a) displays the relationship between the N and the average AoI, while Fig. 3(b) correlates the N with the maximum AoI. The results for each N of the learning-based algorithms shown in Fig. 3 are obtained in the following setup: First, each algorithm is trained, and then the learned policies are applied to a new task, after which 10000 episodes are trained in the new task. The results indicate that both the average and maximum AoI for all algorithms escalate as N increases. The reason for this occurrence is that, as N increases, the UAV takes longer to collect data from all SNs, and the number of ECPs G in the scenario may also increase. These lead to an increase in both average and maximum AoI. Notably, the learning-based methods demonstrate superior performance over heuristic AoI-based algorithms in managing both average and maximum AoI. This arises because learning-based methods are capable of learning long-term cumulative rewards across multiple episodes, in contrast to AoI-based algorithms, which focus solely on the SN with the highest AoI at each decision-making phase. Furthermore, Fig. 3 reveals that the MLDRL-based algorithm exhibits exceptional adaptability to new tasks, achieving the lowest maximum and average AoI among the evaluated algorithms. This advantage is attributed to the fact that MLDRL is able to learn generic knowledge for different tasks during the training phase, thus allowing the proposed algorithm to adapt quickly to new tasks.

V. Conclusions

In this paper, we have studied the challenge of efficiently collecting fresh data from SNs whose locations may change in a UAV-assisted WSN. By segmenting the environment into distinct clusters, we simplify the complexity of the task. Our objective is to optimize the UAV's trajectory to minimize the weighted sum of the average and the maximum AoI across all SNs. We have modeled this problem as an MDP with non-uniform time steps and proposed a MLDRL-based algorithm to solve this problem. This algorithm can learn a generalized policy from a training task set and thus adapt quickly on new tasks. Our simulation results have demonstrated that the MLDRL-based algorithm not only converges more swiftly but also adjusts more promptly to changes in the environment, consistently outperforms the baseline methods across various the number of SNs.

References

[1] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras, "Applications of wireless sensor networks: an up-to-date survey," *Applied system innovation*, vol. 3, no. 1, p. 14, 2020.

[2] N. H. Chu, D. T. Hoang, D. N. Nguyen, N. Van Huynh, and E. Dutkiewicz, "Joint speed control and energy replenishment optimization for uav-assisted iot data collection with deep reinforcement transfer learning," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5778–5793, 2023.

[3] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, 2021.

[4] M. Yi, X. Wang, J. Liu, Y. Zhang, and R. Hou, "Multi-task transfer deep reinforcement learning for timely data collection in rechargeable-uav-aided iot networks," *IEEE Internet Things J.*, 2023.

[5] H. Cao, H. Yao, H. Cheng, and S. Lian, "A solution for data collection of large-scale outdoor internet of things based on uav and dynamic clustering," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9. IEEE, 2020, pp. 2133–2136.

[6] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and Z. Gao, "Uav trajectory planning for aoi-minimal data collection in uav-aided iot networks by transformer," *IEEE Trans. Wireless Commun.*, vol. 22, no. 2, pp. 1343–1358, 2023.

[7] J. Liu, P. Tong, X. Wang, B. Bai, and H. Dai, "Uav-aided data collection for information freshness in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2368–2382, 2020.

[8] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[11] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[12] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a gpu," *arXiv preprint arXiv:1611.06256*, 2016.

[13] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[14] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.