Assisting Natural Beekeeping in Rural and Remote Areas using LoRa-based IoT and Drones
- System Model and Performance Evaluation -

I. System Model and Problem Formulation

Consider a framework consisting of a single UAV and N number of IoTDs with G clusters, as shown in Fig.1, where the UAV visits all clusters. We propose a wireless MEC-enabled UAV-assisted IoT network, comprising a control station s_0, clusters, and IoTDs. The IoTDs (users), denoted as $\mathbb{U} = \{1, 2, \ldots, N\}$, where N represents the total number of IoTDs. These devices are spatially distributed within a defined area in an unevenly distributed manner and are dynamically grouped into clusters to facilitate efficient data management and collection with the consideration of a random walk model for the IoTDs movement.

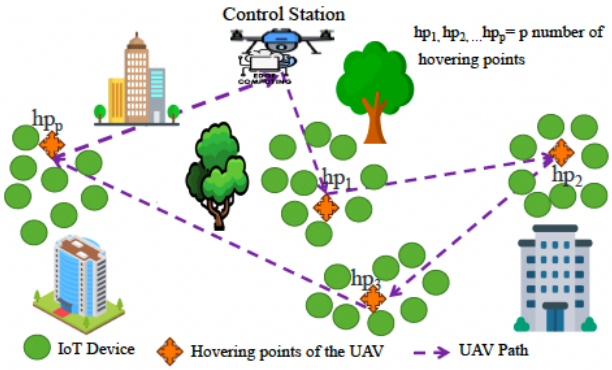


Fig. 1: The wireless MEC-enabled UAV-assisted data collection Model.

The set of clusters at any given time t can be defined as $\mathcal{C}(t) = \{c_1, c_2, \ldots, c_G\}$, where G is the total number of clusters, as shown in Fig. 1. Each IoTD u belongs to exactly one cluster and is represented by the subset $\mathcal{U}_{c_i} \subseteq \mathbb{U}$. Then, the position of the u^{th} IoTD within a cluster c_i at time t, is defined as $w_u^{c_i}(t) = (x_u^{c_i}(t), y_u^{c_i}(t), 0)$.

We define the initial location of the UAV as $s(0) = (X(0), Y(0), Z(h))$. After its departure from the control station, the UAV's position at time t is formulated as $p(t) = (X(t), Y(t), Z(t))$. Its complete trajectory, denoted as $L = \langle \mathbf{H} \rangle$, includes the dynamically determined hover points H within each visited cluster and its start and end at the control the control station. Each hover point identified by the proposed algorithm, $\mathbf{H} = hp_1, hp_2, ..., hp_p$ is associated with a specific cluster c_i. We define the hovering point position at time t in a target cluster as $u^{c_i}(t) = (X^{c_i}(t), Y^{c_i}(t), Z^{c_i}(t))$. Our proposed method identifies the optimal hover points based on the priority metrics of the IoTDs within each cluster. It learns to select IoTDs with higher AoI and data volume and visits first. We formulate that the UAV's flight path should start and end at $s(0)$ not only to maximize data collection capability but also to guarantee the UAV's safe return to the control station.

The UAV hovers over the selected hover points (HPs) within each cluster to collect data from IoTDs. The IoTDs send report messages, including AoI (to verify the generation time of IoDs's data), data volumes, and their locations, to the UAV. Upon reaching a cluster, the IoTDs activate high-speed long-range New Radio (NR) links to transmit their data to the UAV, in which 3GPP Rel. 17 supports communication standards for Cellular-IoT (CIoT) [1]. To this end, it follows a flight path that includes multiple HPs, strategically determined to optimize data collection.

A. Dynamic K-means Clustering Model

Clustering plays a pivotal role in UAV trajectories by reducing travel distance, and communication overhead, and providing efficient path planning. K-means is a widely used clustering algorithm that partitions a dataset into K clusters based on the similarity of data points [2]. Moreover, since the IoTDs are mobile, we use dynamic K-means clustering, which is illustrated in the future section. The algorithm recursively allocates data points to the nearest cluster centroid and updates the centroids based on the mean of the allocated points until convergence is attained. We introduced a binary indicator δ_{w_j,c_i} to represent the link between IoTDs and their allocated clusters. This indicates whether a specific IoTD w_j belongs to cluster c_i, and defined as follows:

$$\delta_{w_j,c_i} = \begin{cases} 1, & \text{if IoTD } w_j \text{ is in cluster } c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We apply the following constraint to ensure that each IoTD is solely designated to only one cluster:

$$\sum_{c_i \in \mathcal{C}} \delta_{w_u, c_i} = 1, \quad \forall w_u \in \mathcal{U} \tag{2}$$

where $\mathcal{C}$ is the set of all clusters, and $\mathcal{U}$ is the set of all IoTDs.

The set of clusters visited by a UAV, denoted by C_{vis}, is tracked as:

$$C_{vis} = \{c_i \in \mathcal{C} \mid \sigma_{c_i} = 1\} \tag{3}$$

where σ_{c_i} is a binary indicator that verifies whether the UAV has visited cluster c_i, which is critical for optimizing the UAV flight routes.

B. Age of Information (AoI)

AoI appears as an important parameter that assesses the freshness of data, which is vital for mission-critical applications like healthcare and telemedicine, traffic management and control, disaster management, and relief operations among others. By prioritizing fresh data collection, the UAV ensures timely updates, crucial for time-sensitive applications [3]. This strategy amplifies decision-making accuracy in dynamic scenarios and enables proactive issue identification within the IoT network. Such a preemptive approach to problem-solving is especially beneficial in rapidly evolving environments.

We represent the AoI as $\text{AoI}(^{c_i}_u)$. It is computed as the time elapsed since the last data update was received from the IoTD. For IoTD u at the current time t in cluster c_i, with $t_{\text{gen},u}$ being the timestamp of the last received data packet. It is given by:

$$\text{AoI}(^{c_i}_u)(t) = t - t_{\text{gen},u} \tag{4}$$

The cumulative AoI $a_t^{c_i}$ which is used to prioritize clusters for UAV visits and helps to identify clusters where the overall data freshness is most lacking is defined as:

$$\text{AoI}^{c_i}(t) = \sum_{u \in c_i} \text{AoI}(^{c_i}_u) \tag{5}$$

To analyze the AoI in general across all clusters, the total AoI a_{total} can be tracked as:

$$\text{AoI}_{total} = \sum_{i=1}^{G} \text{AoI}_t^{c_i} \tag{6}$$

Additionally, the set of AoI values for IoTDs in cluster c_i is represented as:

$$A^{c_i} = \{\text{AoI}(^{c_i}_u) \mid u \in c_i\} \tag{7}$$

When the UAV arrives at a selected cluster, individual IoTDs with higher AoI are prioritized for data updates to guarantee efficient management of information freshness within the cluster. Furthermore, to maintain the relevance and timeliness of the information across the network, we establish an AoI threshold, AoI_{max} as:

$$\text{AoI}(^{c_i}_u) \leq \text{AoI}_{max} \quad \forall u \in \mathcal{U} \tag{8}$$

C. Time Consumption Model

In our scenario, the mission time in our system constitutes transit/flight time, data transmission time, and data computation time.

1) UAV Transit Time:: The UAV flight time or transit time ($T_{tr}^{c_i}$) model computes the time required for the UAV to travel to each cluster starting from the control station. For a given cluster c_i, the transit time is calculated based on the distance from the UAV's current location to the hovering point in a cluster divided by the UAV's constant speed v. It can be computed as:

$$T_{tr}^{c_i} = \frac{d_{cr,c_i}}{v} \tag{9}$$

where d_{cr,c_i} is the Euclidean distance from the UAV's current location to the cluster c_i.

2) Data Transmission Time:: In addition to the flight time, data transmission time (DTT) ($\text{DTT}^{c_i}u$ is another key component. DTT refers to the time it takes for data to be transmitted from the IoTDs to a UAV. It depends on the data volume Vu^{c_i} and the upload rate $R_u^{c_i}$:

$$\text{DTT}(^{c_i}_u) = \frac{Vu^{c_i}}{R_u^{c_i}} \tag{10}$$

Given the UAV's MEC capabilities and its use of wireless communication to collect data from multiple IoTDs simultaneously, the total transmission time for cluster c_i is the sum of the transmission times of all IoTDs within a cluster:

$$T_{DTT}^{c_i} = \sum_{u=1}^{n} \text{DTT}(^{c_i}_u) \tag{11}$$

3) Data Computation Time:: Since we considered that the UAV is equipped with MEC, the UAV processes the data from IoTDs while concurrently collecting. Suppose that all computation tasks from IoTDs in cluster c_i are offloaded to the UAV, the computation time for each IoTD u, denoted as $T_{cp,u}^{c_i}$, depends on the computation workload $c_{r,u}^{c_i}$ and the assigned computing resource f_c. Thus, the computation time is given by:

$$T_{cp,u}^{c_i} = \frac{c_{r,u}^{c_i}}{f_c} \tag{12}$$

Then, the cumulative computation time for cluster c_i is the sum of the computation times of all IoTDs within the cluster:

$$T_{cp}^{c_i} = \sum_{u=1}^{n} T_{cp,u}^{c_i} \tag{13}$$

4) Cluster-wise Time Consumption:: We define T^{c_i} as the total time spent on the UAV's mission in each cluster c_i. The total UAV's mission time in a cluster is the sum of the transit time between clusters ($T_{tr}^{c_i}$), and the Maximum of the total DTT ($T_{DTT}^{c_i}$) and computation time ($T_{cp}^{c_i}$). We use a Max function because the UAV collects and processes data at the same time, allowing us to consider taking the maximum time among the transmission and computation time. In this logic, we show how much time is spent by the UAV in each cluster for transmission and processing. We define it as:

$$T^{c_i} = T_{tr}^{c_i} + max(T_{DTT}^{c_i} + T_{cp}^{c_i}) \tag{14}$$

5) Cumulative Mission Time:: The overall mission time, $T_{mission}$, is the cumulative amount of time the UAV spends completing its tasks across all clusters. It is computed as:

$$T_{mission} = \sum_{i=1}^{G} T^{c_i} \tag{15}$$

where G denotes the total number of clusters. The $T_{mission}$ increases as the number of clusters G increases. From this relation, we noticed that the UAV needs to visit more clusters, resulting in longer transit times and possibly more data transmission and computation time. It is defined as:

$$T_{mission} \propto G \tag{16}$$

D. Optimisation Problem

The main goal of our wireless MEC-enabled UAV-assisted IoT data collection scenario is to optimize the UAV path planning to minimize both the mission time and the AoI across clusters. Meeting these two goals results in timely data collection and processing, which is crucial for mission-critical real-time applications and decision-making processes.

$$\begin{aligned}
\min_{\boldsymbol{H}} \quad & \sum_{i=1}^{G} \left(\alpha T^{c_i}(t) + (1-\alpha)\cdot \mathrm{AoI}^{c_i}(t)\right) \\
\text{s.t.} \quad & \text{C1: } \mathrm{AoI}(_u^{c_i}) \le \mathrm{AoI}_{\max} \quad \forall u \in \mathcal{U}, \\
& \text{C2: } x_{\min} \le x_u^{c_i}(t) \le x_{\max} \quad \forall i, u, \\
& \text{C3: } y_{\min} \le y_u^{c_i}(t) \le y_{\max} \quad \forall i, u, \\
& \text{C4: } \sum_{c \in \mathcal{C}} \delta_{u,c_i} = 1 \quad \forall u \in \mathcal{U}, \\
& \text{C5: } X_{\min} \le X_u^{c_i}(t) \le X_{\max} \quad \forall i, u, \\
& \text{C6: } Y_{\min} \le Y_u^{c_i}(t) \le Y_{\max} \quad \forall i, u, \\
& \text{C7: } Z_{\min} \le Z_u^{c_i}(t) \le Z_{\max}, \\
& \text{C8: } \sigma_{c,u} \le 1 \quad \forall c \in \mathcal{C}, \\
& \text{C9: } \mathrm{hp}_p^0 = s_0,\ \mathrm{hp}_p^{C+1} = s_0.
\end{aligned} \tag{17}$$

where α is a weighting factor between mission time and AoI. Constraint $\mathbf{C_1}$ assures the AoI of the IoTD u is smaller than the maximum threshold. $\mathbf{C_2}$ and $\mathbf{C_3}$ respectively constrained the IoTDs to be within their operational location bounds. Constraint C_4 handles every IoTD should have to Join only one cluster. $\mathbf{C_5}$ and $\mathbf{C_6}$ control the UAV's motions in the X and Y axes. $\mathbf{C_7}$ manages the heights of the UAV between the defined minimum and maximum altitude levels. We apply $\mathbf{C_8}$ to ensure that the UAV visits a cluster only once, and $\mathbf{C_9}$ to command the UAV to start and finish its mission at the initial location.

II. Proposed Solution

We implement an adaptive optimal hovering point selection algorithm for our wireless MEC-enabled UAV-assisted IoT network. Given a cluster of IoTDs c_i, each with distinct AoI and data volume values, the UAV's task is to dynamically select an optimal hovering point **H**. The algorithm dynamically adjusts the hover point by prioritizing IoTDs with a higher AoI and data volume of each IoTD within the cluster. The optimization problem is challenging due to the continuous and complex nature of the UAV's action space, including precise 3D positioning. To overcome the challenges of solving dynamic decision-making problems, we draw inspiration from Deep Learning (DL) and Reinforcement Learning (RL). We reformulate the original problem as an MDP and introduce a new solution based on DDPG, the *[*DDPG-DHPSA]. We also discuss DDPG's foundations and its suitability as a basis for *[*DDPG-DHPSA]. In this section, we first discuss the fundamentals of DDPG and the rationale for its suitability as the basis for the *DDPG-DHPSA* solution.

A. DDPG Primer

The DDPG model, an excellent model-free algorithm in the sphere of DRL, is especially suitable for high-dimensional continuous action continuous action spaces [4]. This approach combines value-based and policy-based methods, making it ideal for optimizing UAV trajectory. In the DDPG, there are four neural networks: the actor network, the critic network, the actor target network, and the critic target network [5]. In the actor network, weights θ_π are periodically updated, current actions a' are selected based on the perceived state s', and s' and R are calculated by interacting with the environment, defined as $\pi, (a = \pi(s|\theta_\pi)$. In response to the current state of the environment, the actor selects an action, which is carefully chosen from a range of available actions. On the other hand, the critic network updates the weight θ_Q of the value network and provides the current Q value, $Q = Q(s, a|\theta_Q)$. It estimates the value of state-action pairs, guiding the actor's decisions. Then, the actor's policy updates evaluate the actions taken by the actor, considering the environment's present state, and send feedback to the policy network for improvement. Using the Bellman equation, the optimal accumulative Q-value of the critic can be defined:

$$Q_*\left(s, a|\theta_Q\right) = \max_{a\in\mathcal{A}, r, s'\sim E}\left[r(s,a) + \gamma \max_{a'} Q_*(s', a'|\theta_Q)\right] \tag{18}$$

r represents the reward obtained from the environment following the execution of action a in state s, $\gamma \in [0, 1]$ represents the discount factor that reduces the significance of subsequent rewards, and s' and a' represent the action to be performed and subsequent state.

The DDPG framework combines the best policy determination and Q-learning techniques. By implementing this, its key objective is for the policy to deterministically link states to their optimal actions to maximize the expected cumulative reward, denoted as $J(s|\theta_\pi)$. The expected cumulative reward is derived based on the state s. According to Q-Learning principles, the critic's learning process relies on the Bellman equation (18). Based on the policy gradient approach, the actor's parameters are adjusted:

$$\begin{aligned}\nabla_{\theta_\pi} J(\theta_\pi) &= \mathbb{E}\left[\nabla_{\theta_\pi} Q\left(s, a|\theta_Q\right)\right] \\ &= \mathbb{E}\left[\nabla_a Q\left(s, a|\theta_Q\right) \nabla_{\theta_\pi}\left(a\right)\right] \\ &= \mathbb{E}\left[\nabla_a Q\left(s, a|\theta_Q\right) \nabla_{\theta_\pi}\left(\pi'(s)\right)\right]\end{aligned} \tag{19}$$

To encourage exploration, DDPG enhances the original actor policy $\pi(s|\theta_\pi)$ by adding noise $\mathcal{N}$, generated by the Ornstein-Uhlenbeck process:

$$\alpha_n = \pi(s|\theta_\pi) + \mathcal{N} \tag{20}$$

We use target actor networks and target critic networks for stabilizing the learning process. Using uniform sampling from the replay buffer, the actor target network θ'_π determines the action a' based on the state s'. As for the critic target network θ'_Q, it calculates the current value Q and derives the target value based on it.

We represented the replay buffer as $\mathcal{R}_\mathcal{B}$ to store experience tuples (s, a, r, s'). The mini-batches $\mathcal{M}_\mathcal{B}$ are sampled from this buffer to update both the value and policy networks. Then, we computed the mean-squared Bellman error (MSBE) loss from the mini-batches. As a result, for the critic network, loss function $L(\theta_Q)$ is calculated as a mean squared error between the predicted Q-values and the target values as:

$$L(\theta_Q) = \mathbb{E}_{s,a,r,s'\sim\mathcal{M}_\mathcal{B}}\left[\left(Q(s, a|\theta_Q) - y\right)^2\right] \tag{21}$$

In this case, $Q(s, a|\theta_Q)$ is the Q-value predicted by the critic network for state s and action a with parameter θ_Q. The expectation $\mathbb{E}$ is taken over the mini-batch of experiences (s, a, r, s') sampled from the replay buffer $\mathcal{M}_\mathcal{B}$. y is the target value derived from the optimal cumulative $Q_*(s, a|\theta_Q))$. Based on the Bellman equation (18), the target value y is computed as:

$$y = r(s, a) + \gamma Q'_{\theta'_Q}\left(s', \pi'_{\theta'_\pi}(s')\right) \tag{22}$$

As defined in equation (21), our objective is to minimize the MSBE loss to ensure the Q-function $(Q(s, a|\theta_Q))$ aligns closely with the target value y. To maintain stability while minimizing the MSBE, we update the parameters of both the target actor

network ($\theta'\pi$) and the target critic network ($\theta'Q$) during each iteration of the main network update cycle as follows with τ as a target network update rate:

$$\theta'_\pi \leftarrow \tau\theta_\pi + (1-\tau)\theta'_\pi$$
$$\theta'_Q \leftarrow \tau\theta_Q + (1-\tau)\theta'_Q \tag{23}$$

B. MDP Formulation

In our system, the UAV operates as the DRL agent, making decisions based on its interactions with the dynamic IoT environment. The UAV's role involves gathering and analyzing data from the network, akin to a central decision-making unit that manages and optimizes the flow of information. To this end, the UAV trajectory optimization problem is formulated as an MDP, defined by the tuple ($\mathcal{S}, \mathcal{A}, p, r, \gamma$) which includes a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function ($r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$), a state transition p and a discount factor $\gamma \in [0, 1]$, respectively.

State Space: The state space $\mathcal{S}$ in the wireless MEC-enabled UAV-assisted IoT network consists of the location of IoTDs and a UAV, the data volume, and the AoI of each device. Each state $s \in \mathcal{S}$, is defined as a vector as:

$$s_t = \langle \mathbb{P}^{c_i}(t), \mathbb{W}_u^{c_i}(t), \mathbb{A}_u^{c_i}(t), \mathbb{V}_u^{c_i}(t) \rangle \tag{24}$$

where, $\mathbb{P}^{c_i}(t) = (X^{c_i}(t), Y^{c_i}(t), Z^{c_i}(t))$ and $\mathbb{W}_u^{c_i}(t) = (x_u^{c_i}(t), y_u^{c_i}(t), 0)$, denote the location of a UAV and the IoTDs, respectively. ($\mathbb{A}_u^{c_i}(t)$ is a vector of the AoI for each IoTD in the cluster, and $\mathbb{V}_u^{c_i}(t)$ denotes the vector of pending data volumes each IoTD in the cluster at time t.

Action Space: By adjusting its strategy to hover point positioning within the chosen cluster for optimal data collection, *DDPG-DHPSA* determines the UAV's next action based on the observed state. At every point in its trajectory, $a \in \mathcal{A}$ represents a decision the UAV takes, allowing it to dynamically alter its flight path and data collection strategy in response to changing environments at time t:

$$a_t = \langle H_u^{c_i}(t) \rangle \tag{25}$$

where $H_u^{c_i}(t)$ is the hovering point within the selected cluster, defined as $H_u^{c_i}(t) = (X_u^{c_i}(t), Y_u^{c_i}(t), Z_u^{c_i}(t)$.

State Transition Probability: The state transition probability acts as a probabilistic indicator for transitioning from one state $s \in \mathbb{S}$ to another $s' \in \mathbb{S}$ after executing action $a \in \mathbb{A}$ in a given system. It quantifies the likelihood of such state changes (with probability $P(s'|s, a)$. Considering the Markov property, the state transition probability is defined as:

$$P(s'|s,a) = \mathbb{P}^{c_i}(s'), \mathbb{W}_u^{c_i}(S'), \mathbb{A}_u^{c_i}(s'), \mathbb{V}_u^{c_i}(s')$$
$$|\mathbb{P}^{c_i}(s), \mathbb{W}_u^{c_i}(s), \mathbb{A}_u^{c_i}(s), \mathbb{V}_u^{c_i}(s)) \tag{26}$$

Reward Function: The reward function $r : S \times A \rightarrow \mathbb{R}$ encourages actions that reduce overall flight time and AoI while ensuring efficient data collection from IoTDs. UAV actions are quantified to determine how effective they are in achieving these objectives. Actions that reduce AoI and increase data collection time are incentivized more, while actions that increase AoI and prolong mission duration are penalized. It is the agent's responsibility to identify an optimal policy π_* that maximizes the expected reward. As a result, the Q-value can be described as:

$$Q(s, a|\theta_Q) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1}(r|s, a)\right] \tag{27}$$

where T is the number of time slots and the optimal policy is defined as:

$$\pi_* = \arg\max_{\pi} Q(s, a|\theta_Q), \quad \forall s \in \mathcal{S} \tag{28}$$

The reward r is defined as the negative of the weighted sum of time and AoI in a cluster-wise manner:

$$r_t(s_t, a_t) = -\sum_{i=1}^{G} (\alpha T^{c_i}[t] + (1-\alpha) \cdot AoI^{c_i}[t] + \xi + P) \tag{29}$$

Where ξ is a coverage bonus reward to encourage the UAV to successfully cover all devices in a cluster and the entire cluster. The UAV gets a relatively large penalty P when it does not meet the constraints set in 17. P is composed of penalties for critical conditions P_c and operational constraints P_o. P_c, the penalty associated with critical conditions, is calculated as $P_c = \gamma_c \cdot n_c$, where γ_c is the penalty coefficient per device that exceeds critical thresholds for AoI, and n_c is the number of such devices. Operational constraints P_o are penalized by $P_o = \gamma_o$ if violated, and by 0 otherwise. Then, the overall penalty P is given by $P = P_c + P_o$.

In our wireless MEC-enabled UAV-assisted systems, the dynamic K-means algorithm is utilized to shorten the UAV's flight length as defined in Algorithm **??**. It iteratively assigns IoTDs to the nearest cluster centroid and updates centroids until convergence ($\Delta < \varepsilon$, where $\varepsilon \approx 0.01$). After each convergence, it assesses environmental changes (Δ_{env}) between time steps,

including IoTD movements and additions or removals. If significant changes occur ($\Delta_{env} > \delta$, where δ is typically set to 10 meters, the clustering process is reinitialized. Due to its simplicity and resource-efficient mobility, we assume the Random-Walk Mobility (RWM) type, which is well-suited for our scenarios and common in many IoT scenarios in real-world applications [6]. Moreover, random-walk best represents the unpredictable nature of many IoT deployments, such as in smart cities and industrial IoT settings.

Hence, the *DDPG-DHPSA* algorithm is strategically designed to minimize overall mission time and AoI. For detailed pseudo-code, refer to Algorithm 2. In addition, as shown in the diagram 2, the architecture and workflow of the *DDPG-DHPSA* algorithm is illustrated.

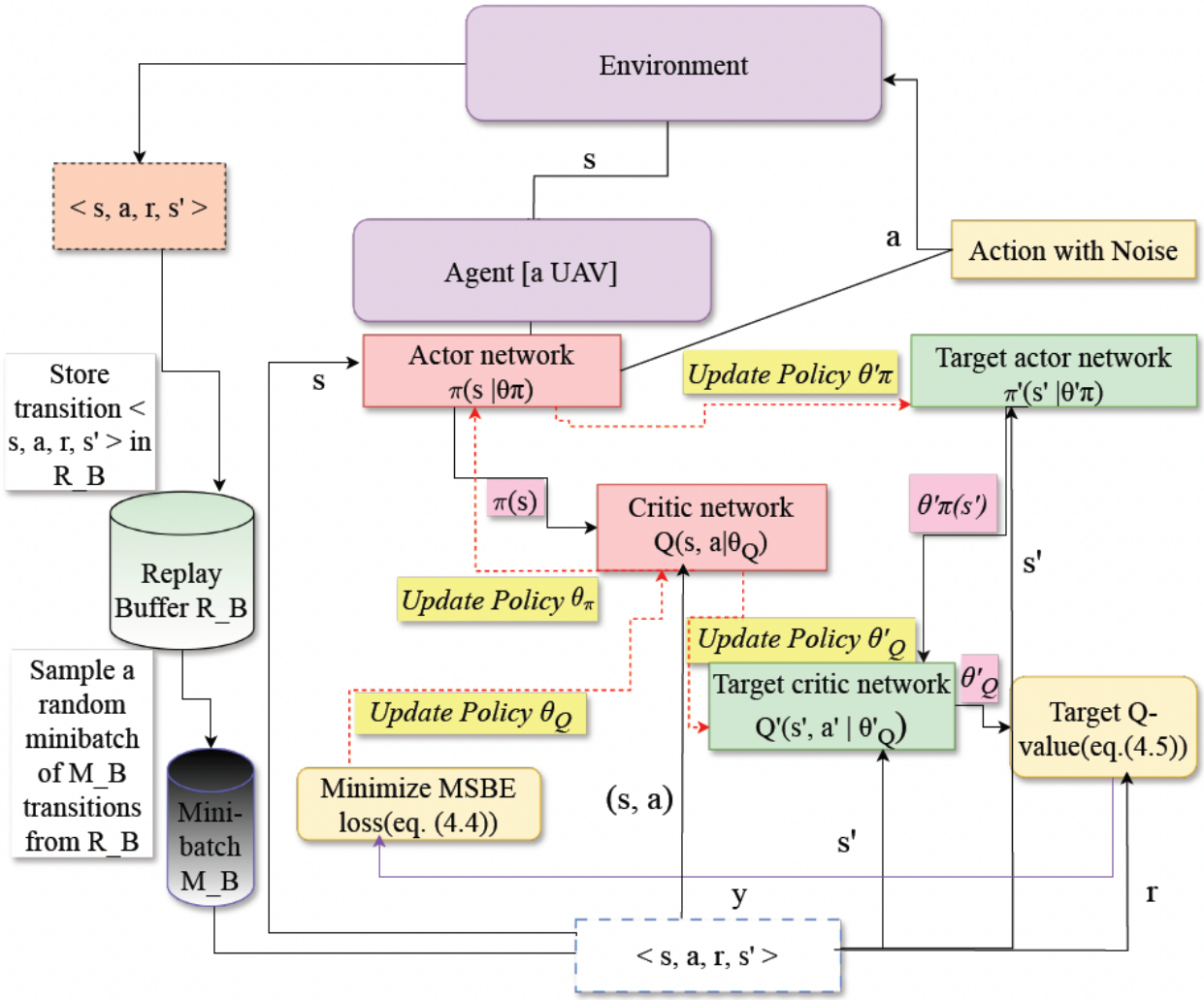


Fig. 2: *DDPG-DHPSA* algorithm.

III. Simulation Results

A. Simulation Environment and Setup

Our simulation uses an NVIDIA GeForce GTX 1650 GPU and an Intel(R) Core(TM) i5-9400 CPU. Python 3.10.13 (64-bit) and PyTorch 1.13.1 with CUDA 11.6 support and Gym 0.26.2 are used to program the software environment. In these experiments, we carefully consider hyperparameters and training settings and record the performance measures for analysis, and evaluation along with the training progress.

In the simulation, we assume the UAV is collecting data from 100 IoTDs in a 200 × 200 m area at a constant speed of 15 m/s, which aligns with the optimal maneuver speed for our configuration. The UAV flies at the altitudes between 60 and 300 m. For more details, Table I lists other environmental parameters. We set two hidden neural network layer values with 512 and 256 nodes, and the batch and buffer sizes with 16 and 50000, respectively. 0.01 for the soft update coefficient and 0.99 for the discount factor. We compare the proposed algorithm with four different algorithms:

- **Proposed**: This is the main framework being evaluated. It integrates three key components: (1) a K-Mean algorithm for clustering the IoTDs based on their spatial proximity, (2) a DDPG algorithm for optimizing the hovering points of the UAV

Algorithm 1 Dynamic K-means Clustering

Input: Initial number of clusters N_{init}, convergence threshold $\varepsilon > 0$, change threshold $\delta > 0$, N_{min}, N_{max}, UAV speed $v = 15$ m/s, time step Δt, min altitude Z_{min}, max altitude Z_{max}, initial UAV position $s(0) = (X(0), Y(0), Z(h))$
Output: Cluster assignments $\{C_1, \ldots, C_N\}$ at each time step t

$N \leftarrow N_{init}$
Initialize centroids $\{c_1, \ldots, c_n\}$ randomly in $\mathbb{R}^3$
$p(0) \leftarrow s(0)$
$step\_size \leftarrow v \cdot \Delta t$
$t \leftarrow 0$
loop
  repeat
    $C_i \leftarrow \emptyset, \forall i \in \{1, \ldots, N\}$
    for $w_j \in \mathcal{J}_t$ **do** ▷ $\mathcal{J}_t$ is the set of IoTDs at time t
      $i^* \leftarrow \arg\min_{i \in \{1,\ldots,N\}} \|w_j - c_i\|_2$
      $C_{i^*} \leftarrow C_{i^*} \cup \{w_j\}$
    end for
    $c_i' \leftarrow \frac{1}{|C_i|} \sum_{w_j \in C_i} w_j, \forall i \in \{1, \ldots, N\}$
    $\Delta \leftarrow \max_{i \in \{1,\ldots,K\}} \|c_i' - c_i\|_2$
    $c_i \leftarrow c_i', \forall i \in \{1, \ldots, N\}$
  until $\Delta < \varepsilon$
  $\Delta X \sim U[-1, 1]$
  $\Delta Y \sim U[-1, 1]$
  $\Delta Z \sim U[-1, 1]$
  Normalize $(\Delta X, \Delta Y, \Delta Z)$ to unit vector
  $X(t+1) \leftarrow X(t) + \Delta X \cdot step\_size$
  $Y(t+1) \leftarrow Y(t) + \Delta Y \cdot step\_size$
  $Z(t+1) \leftarrow Z(t) + \Delta Z \cdot step\_size$
  $Z(t+1) \leftarrow \max(Z_{min}, \min(Z(t+1), Z_{max}))$
  $p(t+1) \leftarrow (X(t+1), Y(t+1), Z(t+1))$
  $\Delta_{env} \leftarrow \sum_{w_j \in \mathcal{J}_t \cap \mathcal{J}_{t+1}} \|w_{j,t+1} - w_{j,t}\|_2 + |\mathcal{J}_{t+1}| - |\mathcal{J}_t| + \|p(t+1) - p(t)\|_2$
  if $\Delta_{env} > \delta$ **then**
    $WCSS \leftarrow \{\}$
    for $n \in \{N_{min}, \ldots, N_{max}\}$ **do**
      Run K-means with n clusters on $\mathcal{J}_{t+1} \cup \{p(t+1)\}$
      $WCSS[n] \leftarrow \sum_{i=1}^{n} \sum_{w_j \in C_i} \|w_j - c_i\|_2^2$
    end for
    $N_{new} \leftarrow$ ElbowMethod($WCSS$) ▷ Find elbow point
    if $N_{new} \neq N$ **then**
      $N \leftarrow N_{new}$
      Reinitialize centroids $\{c_1, \ldots, c_N\}$ randomly
    end if
  end if
  $t \leftarrow t + 1$
end loop

TABLE I: Environmental Parameters

Parameter	Value
β_0	30 (dB)
α	2
N_0	-174 (dBm/Hz)
Bandwidth	1 MHz
p_j	20 dBm
V_j	0.5 $\sim$ 1 Mbits
Data generation timestamp	0.1 $\sim$ 0.3 s

within each cluster, and (3) an exhaustive search algorithm to find the optimal trajectory of the UAV across the clusters. Combining these techniques, the proposed framework aims to efficiently collect data from IoTDs while minimizing the AoI and mission time.

- **WoCluster**: Without clustering (WoCluster) acts as a baseline comparison in which IoTD clustering is not carried out.

Algorithm 2 *DDPG-DHPSA*

Input: The environment parameters: $\mathbb{P}_u^{c_i}[t]$, $\mathbb{W}_j^{c_i}[t]$, $\mathbb{A}_j^{c_i}[t]$, $\mathbb{V}_j^{c_i}[t]$; *DDPG-DHPSA* training parameters: $E, T_s, \mathcal{R}_\mathcal{B}, \mathcal{M}_\mathcal{B}, \alpha_{actor}, \alpha_{critic}, \gamma, \tau, \alpha_n$.
Output: The trained weights of θ of actor networks for the agent.

```
Initialize the weights θ_π and θ_Q of DNN actor network π(s) and critic network Q(s), respectively.
Initialize the weights θ_π and θ_Q of DNN actor network π(s) and critic network Q(s), respectively.
Initialize target weight θ'_π ← θ_μ of the target actor π' and the target weight θ'_Q ← θ_Q of the target critic Q'.
Initialize replay buffer size R_B and mini-batch M_B. Initialize a Gaussian process with mean = 0 and var = 2.
for episode ← 1 to E do
    Initialize state s based on (24)
    while t ≤ T_s do
        Observe state s
        Actor generates action a according to (25)
        Compute reward r according to (29)
        Observe next state s'
        Store tuple (s, a, r, s') in buffer R_B
        Sampling batch M_B tuples from R_B to train DNN
        Update θ_Q by minimizing loss in (21)
        Update θ_π from policy gradient in (19)
        Update target networks based on (23)
        Increment t
    end while
end for
```

Rather, the DDPG method is implemented to identify the best hovering point while considering all IoTDs. The UAV then heads to a single hovering position to gather data from all IoTDs. This approach provides an understanding of how clustering impacts system performance.

- **WoDDPG**: Without DDPG (WoDDPG), under this approach, the hovering points are established at the center of each cluster without using the DDPG algorithm to optimize. The UAV visits each cluster center to collect data from the IoTDs in that cluster. This method assesses the advantages of implementing DDPG for optimizing hovering locations versus a simpler approach of employing cluster centers.
- **PathGreedy**: We apply the greedy technique described in [7, Sec. III-B] to reduce the complexity of the exhaustive search for determining the trajectory of the UAV. The closest cluster to the UAV's present location is the next cluster to be visited. While this approach offers a computationally efficient way to plan trajectories in real time, a globally optimal solution cannot be guaranteed, which is not the case in the proposed algorithm.
- **KRandom**: This approach selects the hovering locations and the trajectory of the UAV at random after clustering the IoTDs using the Kmean algorithm. This strategy serves as a baseline for comparing the suggested framework's performance in terms of optimizing UAV positioning and path planning against a random method.

B. Proposed Framework Analysis

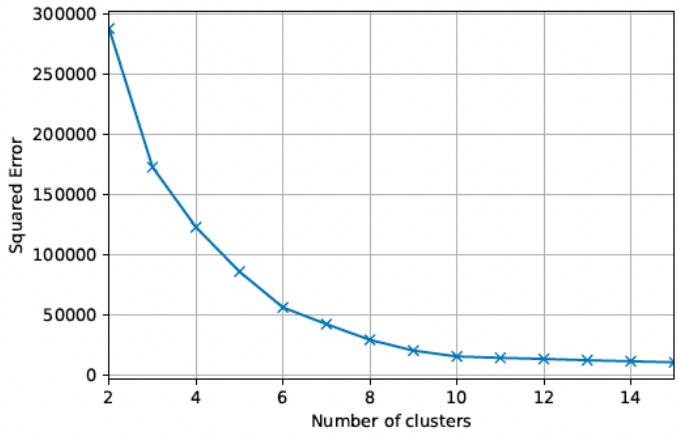
We first cluster them by implementing the dynamic k-mean algorithm to cluster them. This approach adapts to the evolving distribution of IoTDs in real time. We apply continuous monitoring of the positions of IoTDs given that the devices are dynamic. When changes are detected, instead of re-clustering the entire dataset; the algorithm will assign new IoTDs to the nearest cluster and update the centroid, remove the removed IoTDs from their cluster and recalculate the centroid, and reassign the moved IoTDs if necessary and update affected centroids. To optimize the number of clusters in the data, we estimate the loss function, which is obtained by adding the squared errors between cluster centroids and their data points for scenarios ranging from 2 to 15 clusters.

Fig. 3a illustrates the squared error decreases according to the increase in the number of clusters. The elbow method is used to determine the optimal number of clusters, which is determined to be ten.

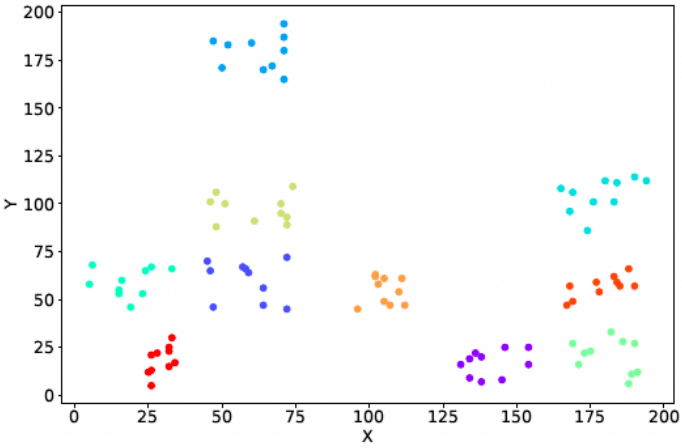
Fig. 3b shows the clustering results, with each cluster represented by distinct colors. This visualization effectively depicts the clustering structure and the distribution of IoTDs among the different clusters.

To figure out the optimal hovering point at each cluster, we utilize the *DDPG-DHPSA* algorithm. Fig. 4 shows the algorithm's training convergence under various learning rate configurations. We explore different actor (r_a) and critic (r_c) learning rates, selecting values from $\{1e^{-3}, 5e^{-4}, 2e^{-4}\}$.

The training results reach convergence after approximately 250 episodes across all scenarios, with the case where $r_a = r_c = 5e^{-4}$ demonstrating the highest outcome. In this manner, to optimize the hovering points for each cluster, we use the model trained in this best-case scenario. Fig. 5, as a result, shows the optimal hovering points of all clusters. Once all hovering points are collected, we optimize the trajectory of the UAV.



(a) K-Mean error according to number of clusters



(b) Clustering results

Fig. 3: Clustering IoTDs using K-Mean

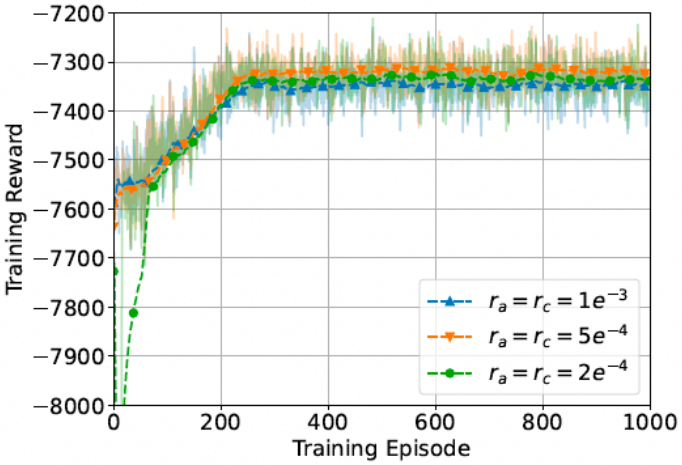


Fig. 4: Convergence of *DDPG-DHPSA* algorithm

To determine the most optimal trajectory, we employ the exhaustive search methodology, which ensures that all feasible cases are considered and explored. Fig. 6 displays the 2D optimal trajectory.

C. Performance Evaluation

In this subsection, we evaluate the system's performance in various environmental conditions.

Fig. 7 depicts the relationship between data volume and AoI performance for the proposed algorithm, WoDDPG, KRandom, WoCluster, and PathGreedy. The results demonstrate that AoI scales linearly with increasing data volume across all five approaches. This proportional growth in AoI can be attributed to the fact that larger data volumes require more time to transmit,

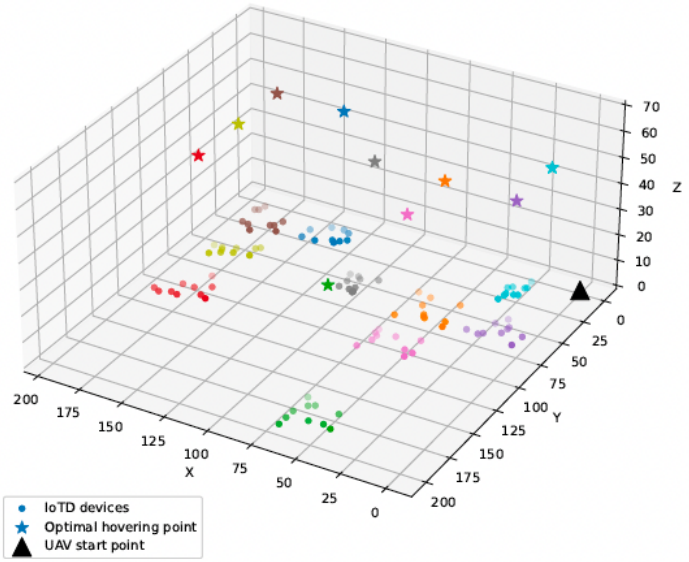


Fig. 5: Hovering points according to IoTD clusters

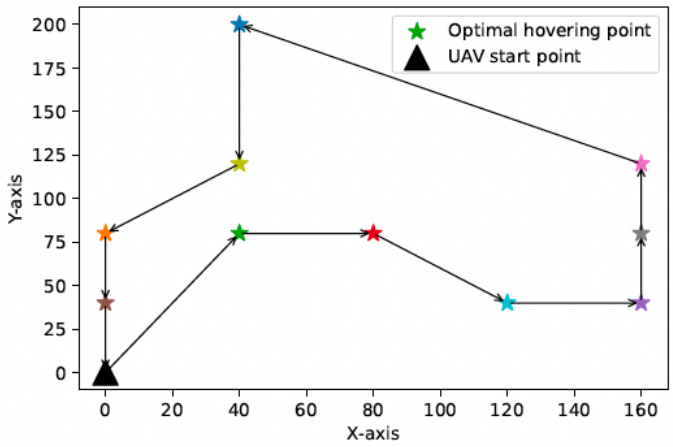


Fig. 6: Optimal trajectory of UAV

leading to older information being received at the destination. Notably, when examining the proposed algorithm, the AoI exhibits an 11% relative increase as the data volume grows by 0.1 MB. This substantial degradation in information freshness underscores the critical impact that data size has on the timeliness of delivered content in the system. The comparative analysis among the five algorithms highlights that the proposed scheme consistently achieves a lower AoI across the tested range of data volumes. This suggests that the optimizations employed in the proposed method are effective at mitigating the AoI penalty incurred by increased data size, offering superior performance in maintaining the novelty of transmitted information.

Accordingly, the proposed method has a better performance than *PathGreedy*, *WoDDPG*, *WoCluster*, and *KRandom* schemes by roughly 8.2%, 73%, 87.5%, and 132%, respectively. Based on the numerical results, we found that our multi-stage approach is effective. The result implies that by clustering IoTDs and applying the *DDPG-DHPSA* algorithm, system performance is improved by 87.5% and 73%, respectively.

Fig. 8 illustrates the impact of varying data generation intervals on the AoI performance for the proposed scheme and the other four comparative approaches. The data generation timestamps at the IoTDs are varied from 0.1 to 0.3 seconds to investigate the effect of delayed data generation on the system's AoI. As demonstrated by the results, the AoI exhibits a decreasing trend as the data generation intervals become longer. The reason the inverse relationship observed in the graph is attributed to the fact that having less frequency of data generation would reduce the time data spends waiting in the queue at the IoTDs to be sent to the UAV. Therefore, the data experiences reduced waiting times, leading to a lower AoI at the destination. Based on our quantitative analysis, we observe that the proposed scheme significantly improves AoI performance, with an average reduction of 4.91 % for every 0.5-second increment of the data generation interval. This significant enhancement highlights the effectiveness of the proposed approach in minimizing the AoI and ensuring the timely delivery of information from the IoTDs to the UAV.

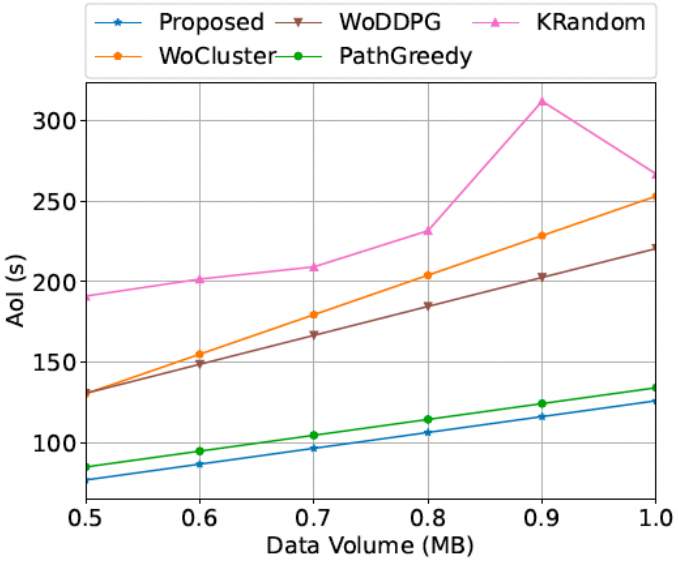


Fig. 7: Performance according to data volume

As shown across the entire range of tested data generation intervals, our approach consistently achieves the lowest AoI values compared to WoDDPG, KRandom, WoCluster, and PathGreedy. A significant reason for this stellar performance lies in the optimized strategies employed in the proposed scheme, which optimize data transmission processes and prioritize the freshness of information delivered. These results highlight how crucial it is to take data generation patterns into account.

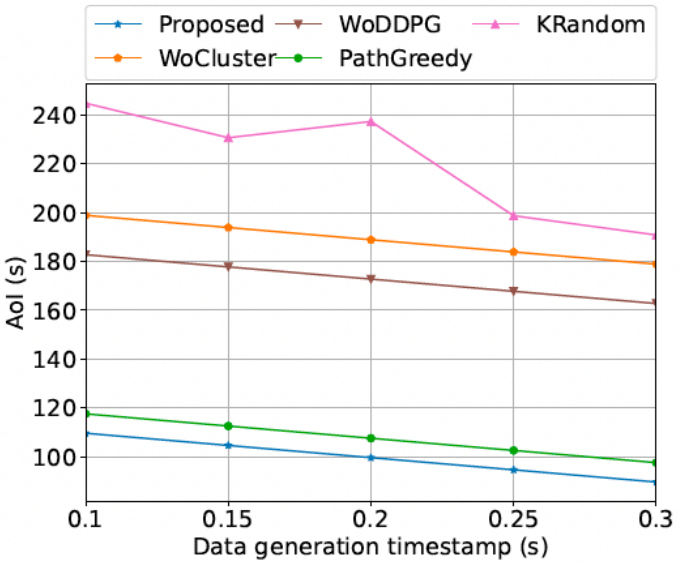


Fig. 8: Performance according to data generation timestamp

REFERENCES

[1] H. Yin, N. Li, J. Guo, J. Zhu, and X. She, "NR Coverage Enhancements for PUSCH," *IEEE Communications Magazine*, vol. 60, no. 7, pp. 36–42, 2022.
[2] K. P. Sinaga and M.-S. Yang, "Unsupervised K-means clustering algorithm," *IEEE access*, vol. 8, pp. 80716–80727, 2020.
[3] Z. Wei, M. Zhu, N. Zhang, L. Wang, Y. Zou, Z. Meng, H. Wu, and Z. Feng, "UAV-assisted data collection for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15460–15483, 2022.
[4] K. Ikeagu, Y. Ding, C. Song, and M. R. Khandaker, "Intelligent Reflecting Surface Optimization for MIMO Communication Using Deep Reinforcement Learning," in *2023 31st Telecommunications Forum (TELFOR)*. IEEE, 2023, pp. 1–4.
[5] Spinning Up OpenAI, "Deep Deterministic Policy Gradient ," https://spinningup.openai.com/en/latest/algorithms/ddpg.html, (Accessed on October 05, 2023).
[6] B. Safaei, A. Mohammadsalehi, K. T. Khoosani, S. Zarbaf, A. M. H. Monazzah, F. Samie, L. Bauer, J. Henkel, and A. Ejlali, "Impacts of mobility models on rpl-based mobile iot infrastructures: An evaluative comparison and survey," *IEEE access*, vol. 8, pp. 167779–167829, 2020.
[7] T. P. Truong, T. M. T. Nguyen, T.-V. Nguyen, N.-N. Dao, and S. Cho, "RSMA for Uplink MIMO Systems: DRL-Based Achievable System Sum Rate Maximization," in *2023 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2023, pp. 878–883.