UAV-Aided Lifelong Learning for AoI and Energy Optimization in Non-Stationary IoT Networks

Zhenzhen Gong, Omar Hashash, *Graduate Student Member, IEEE,* Yingze Wang, Qimei Cui, *Senior Member, IEEE,* Wei Ni, *Fellow, IEEE,* Walid Saad, *Fellow, IEEE,* and Kei Sakaguchi, *Senior Member, IEEE*

***Abstract*—In this paper, a novel joint energy and age of information (AoI) optimization framework for IoT devices in a *non-stationary* environment is presented. In particular, IoT devices that are distributed in the real-world are required to efficiently utilize their computing resources so as to balance the freshness of their data and their energy consumption. To optimize the performance of IoT devices in such a dynamic setting, a novel *lifelong reinforcement learning (RL)* solution that enables IoT devices to continuously adapt their policies to each newly encountered environment is proposed. Given that IoT devices have limited energy and computing resources, an unmanned aerial vehicle (UAV) is leveraged to visit the IoT devices and update the policy of each device sequentially. As such, the UAV is exploited as a mobile learning agent that can learn a shared knowledge base with a feature base in its training phase, and feature sets of a zero-shot learning method in its testing phase, to generalize between the environments. To optimize the trajectory and flying velocity of the UAV, an actor-critic network is leveraged so as to minimize the UAV energy consumption. Simulation results show that the proposed lifelong RL solution can outperform the state-of-art benchmarks by enhancing the balanced cost of IoT devices by 8.3% when incorporating warm-start policies for unseen environments. In addition, our solution achieves up to 49.38% reduction in terms of energy consumption by the UAV in comparison to the random flying strategy.**

***Index Terms*—Internet of Things (IoT), Unmanned Aerial Vehicle (UAV), Age of Information (AoI), Lifelong Learning.**

I. Introduction

THE Internet of Things (IoT) [1] represents a technological breakthrough that brings forth numerous opportunities for new applications at the intersection of wireless communications and intelligent industries, e.g., Industry 4.0 [2]. In fact, IoT devices can provide increased autonomy to physical systems by harnessing their immense capabilities to sense and collect data from their surroundings [3]. In essence, capturing the performance of IoT devices in such use cases has been an active area of research and recent interest [4]. Essentially, this requires adopting novel metrics such as the *age of information (AoI)* to reflect the timeliness and freshness of the underlying physical systems being monitored [5]. Here, the AoI is defined as the time elapsed since the last successfully received update packet at the IoT device was generated by the physical source [6]. Nevertheless, IoT devices often lack sufficient computing capabilities and have limited energy resources to offload their collected data to remote base stations (BSs). Hence, it is challenging for IoT devices to operate in areas with poor connectivity and provide reliable services for mission-critical physical systems. To alleviate such challenges, one can integrate IoT networks with unmanned aerial vehicles (UAVs) [7]–[9] to improve wireless connectivity and enhance computing abilities. UAVs can be deployed as flying BSs that can communicate with IoT devices in a cost-efficient way. It can significantly extend the communication distance, overcome terrain constraints, and enhance communication quality [10]. Additionally, UAVs can augment wireless communication capabilities significantly through the integration of complementary technologies, such as intelligent reflecting surfaces (IRS) [7]. Evidently, UAVs can enhance the performance of IoT networks by providing versatile wireless [8] and computing services [9] to aid autonomous decision-making at the IoT level, and simultaneously, reduce the energy cost of IoT devices.

Despite the surge on IoT in the literature [4], [11], [12], the physical environment associated with IoT networks have been extensively considered stationary – *an ideal assumption that rarely holds in practice* [13]. This assumption normally considers that the generation of data from the physical system follows a given distribution [14] Consequently, the optimal strategies and policies that govern the operation of IoT devices are assumed implicitly to be time-invariant [15]. On the contrary, due to changes in the environment affecting the physical system (e.g., thermal drifts) or on the system level itself (e.g., aging effects), *non-stationary* conditions arise for IoT devices [16]. Thus, it remains challenging for IoT devices to optimize their performance and reduce their AoI and energy costs in such real-world scenarios [17]. Noticeably, relying on conventional methods, such as reinforcement learning (RL), to do so by optimizing the operating policies of these devices drastically fails to address the challenges posed by non-stationary environments. This stems from the fact that such solutions are theoretically developed and tailored towards operating in stationary environments, whereby any variabilities in the environment can lead to suboptimal performance and degradation in the system reliability levels. Henceforth, a robust RL solution that can *continuously adapt the policies* of IoT devices to unprecedented *non-stationary* developments

O. Hashash and W. Saad were supported by the U.S. National Science Foundation under Grant CNS-2210254. K. Sakaghuci was supported by the NICT JUNO project, under grant 22404.

Zhenzhen Gong and Qimei Cui are with the National Engineering Laboratory for Mobile Network Technologies, Beijing University of Posts and Telecommunications, Beijing 100876, China (email: {gzz0822, cuiqimei}@bupt.edu.cn).

Omar Hashash and Walid Saad are with Wireless@VT, Bradley Department of Electrical and Computer Engineering, Virginia Tech, Arlington, VA, 22203, USA (email: {omarnh, walids}@vt.edu).

Yingze Wang is with China Academy of Information and Communications Technology, Beijing 100033, China (email: wangyingze@caict.ac.cn)

Wei Ni is with Data61, CSIRO, Marsfield, NSW 2122, Australia (email: Wei.Ni@data61.csiro.au)

Kei Sakaguchi is with Department of Electrical and Electronic Engineering, Tokyo Institute of Technology, Tokyo, 152-8550, Japan (email: sakaguchi@mobile.ee.titech.ac.jp).

in the environment is desired.

A. Prior Works

Minimizing the AoI has been extensively addressed in various IoT network scenarios [18]–[20] with the aid of UAVs [21], [22]. In particular, the works in [21] and [22] investigate minimizing the AoI for IoT scheduling updates, while optimizing the trajectory [21] and velocity [22] of the UAV. Nevertheless, these works [19]–[21] rely on classical RL approaches (e.g., Q-learning) throughout their solutions. Accordingly, this limits the novelty of these solutions to ideal stationary scenarios. Furthermore, such stationary assumption also affects the practical UAV functionality in assisting IoT networks. In practice, leveraging RL solutions (e.g., value-based learning [23] and policy-based learning [24]) to optimize the UAV operations and trajectory in such non-stationary scenarios can lead to unstable rewards while draining computing resources. To address this issue, recent works such as [25] and [26], have discussed RL solutions for a broad range of applications in a non-stationary setting. Nevertheless, the works in [25] and [26] have key limitations that hinder their practicality. In fact, these works assume that the evolution of non-stationary environments is predictable in nature. Thus, these works assume full knowledge about the period of each environment encountered and its transitions. For instance, the work in [25] models the non-stationary environments as continually evolving Markov decision processes (MDPs) that are based on a soft actor-critic (AC) architecture. While such approach [25] works explicitly for *periodically* changing non-stationary environments, it is clearly not suitable for capturing the *abrupt, random, and miscellaneous* transitions that occur in dynamic real-world environments. Moreover, the variants of Q-learning emerging in [26] have shown inefficient performance on multiple fronts. On the one hand, the previously learned parameters and optimal policies are *discarded* for each new environment encountered. On the other hand, adapting the policies can be computationally intense and resource draining until *convergence* is reached. Furthermore, merging RL with other fundamental methods that provide a leap into the non-stationary realm has also been limited. Here, one of these prominent solutions can be to embrace RL with meta-learning [27] and transfer learning. In particular, this solution can considerably maintain adequate performance in new environments with the aid of a few gradient updates and k-shot learning. Nonetheless, it tends to *lose the knowledge* acquired about previously encountered environments with each new update. In fact, harnessing this knowledge can be a key to enable swift updates [28]. Indeed, exploiting other methods that fill in this gap and facilitate maintaining continuous knowledge transfer (e.g., through a knowledge base [29]) between environments in a RL regime can provide substantial enhancements [25]–[27]. However, they come at the expense of *changing the architecture* of the knowledge base for each new environment encountered, which can be largely inefficient.

In essence, an effective RL solution in a non-stationary setting demands a rigorous design *to detect transitions between environments, accumulate the acquired knowledge and successfully transfer it between environments, while ensuring a minimal convergence time for updates and the sustainability of acquired parameters and architectures.* To this end, this novel RL solution must generalize between environments on the fly, while efficiently utilizing computing and energy resources of both the UAV and IoT devices, simultaneously. Hence, such a solution should consider the optimization of the UAV inherently within its design, as the non-stationary dimension in the IoT network will impinge on the operating strategy and functionality of the UAV. As such, fulfilling these requirements demands a shift towards *lifelong learning* [30] that can provide a *continuously evolving, knowledge-aware, and generalizable* RL solution in a non-stationary setting.

B. Contributions

The main contribution of this paper is the development of a novel UAV-aided *lifelong RL* approach to continuously optimize the data freshness and energy efficiency of IoT devices in non-stationary environments. In particular, IoT devices distributed in the real-world must effectively utilize their computing resources to balance their cost of AoI and energy consumption. This is carried out in the presence of a UAV that facilitates the rapid adaptation of these IoT devices to their dynamic environments through updating their operating policies accordingly. Here, the UAV acts as a central mobile agent responsible for visiting each IoT device while updating and learning a knowledge basis and feature basis of the encountered environments, respectively. Thus, our proposed lifelong approach further focuses on developing a generalizable model that can seamlessly adapt to new environments without altering the underlying knowledge base. In fact, such model with mapping vectors for each environment individually enables the separation of the shared knowledge and environment specific knowledge. Hence, this enables the rapid adaptation to new environments by alleviating the need for extensive modifications or retraining of the model. To efficiently utilize the energy resources of the UAV in this operation, an AC network is leveraged to optimize its flying trajectory and velocity between the devices. By utilizing the accumulated knowledge basis and the extracted features, the framework can effectively determine the best IoT interacting policies for different environments without the need for explicit training on every specific environment. *To the best of our knowledge, this is the first work that considers the joint optimization of both IoT devices and UAV in non-stationary environments.*

In summary, our key contributions include:

- We propose a UAV assisted IoT framework that enables the UAV to help IoT devices adjust to non-stationary environments while minimizing its flying energy. In this framework, the energy consumption of the UAV and the cost of the IoT devices (represented in terms of AoI and computing energy) are jointly considered. A fix and optimize method is applied for better analysis.
- We propose a novel lifelong RL approach that enables knowledge transfer for a stream of environments. By employing this approach, a small amount of sampled data can facilitate convergence within a few steps. In addition, the use of zero-shot method enables the quick feature

extraction such that a warm-start initial policy can be provided for unseen environments.
- A novel environment discovery method is explored to detect the environment change point. Using the sampled information, the application of feature extraction and a doubled knowledge basis together can enable fast knowledge extraction for unseen environments.

Simulation results showcase that our proposed lifelong RL solution can reduce up to 50% of the convergence time for updates in comparison to random initial policies. Moreover, it can reduce the energy consumption of the UAV by 49.38% in comparison to the random flying strategy. Furthermore, in our previous work [1], we have developed a lifelong RL solution to optimize of the AoI and computing energy of IoT devices in a non-stationary setting. However, this early work did not consider the influence of the non-stationary environment on the UAV flying strategy. In contrast, this work comprehensively addresses the flight strategy of the UAV and inherently incoporates it within our lifelong RL approach.

The rest of the paper is organized as follows. The system model and problem formulation are provided in Section II. The lifelong RL algorithm to optimize IoT devices in non-stationary environments is presented in Section III. The design of an energy efficient solution to optimize the trajectory and velocity of the UAV is presented in Section IV. Simulation results are provided in Section V. Finally, conclusions are drawn in Section VI. The notations used in this paper are shown in Table I.

Table I: Parameters and notations of the system model.

Parameters	Descriptions
N	Number of IoT Devices
$\mathcal{N}$	The set of IoT Devices
i	The index of device
t	The index of time slot
$t^s_{i,t}$, $t^e_{i,t}$	Starting and ending time of the environment
$\boldsymbol{d}_i$	The Cartesian coordinates of device i
$\lambda_{i,t}$	Probability of the Bernoulli distribution
$q_{i,t}$	The indicator of the package arrival
$a_{i,t}$	The size of the data packet
$\bar{a}_{i,t}$	Mean value of data package size
$\sigma_{i,t}$	Standard deviation of Gaussian distribution
$\epsilon_{i,t}$	Number of CPU cycles for device i at time slot t
$\epsilon_{i,\max}$	Maximum number of CPU cycles for device i
κ_i	Device's chip architecture related parameter
$b_{i,t}$	Queues of unprocessed packages in CPU cycles
$\Delta_{i,t}$	AoI of device i at time slot t
$\xi_{i,t}$	Index of the most recently processed data packet
$u_{\xi_{i,t}}$	The generation time of packet $\xi_{i,t}$
$\omega_{i,t+1}$	Indicator of the empty queue for time slot t
$c_i(t)$	Cost of device i at time slot t
β, μ, η_x	Trade-off factors with $x = 1, 2, 3, 4$
$z_{i,t}$	Environment specific set of device i at slot t
$\boldsymbol{l}_0$	UAV's initial location
$\boldsymbol{l}_m$	UAV's location in its $m-$th decision
v_m	Velocity of the UAV for its $m-$th flight
$v_{\min}, v_{\max}$	UAV's minimum and maximum velocity
$\boldsymbol{\varepsilon}_i$	The CPU decision vector of device i
$\boldsymbol{E}$	The CPU decision matrix of all the devices
$\boldsymbol{v}$	The vector of UAV's velocity during flight
$\boldsymbol{F}$	The vector of UAV's destination during flight
ϵ_z	Gaussian noise of stochastic policy
σ_z	Standard deviation of Gaussian distribution
e_0	Propulsion energy per unit time
e^U_m	Total energy consumption of UAV's $m-$th flight
$\boldsymbol{\tau}$	Interaction history collected from environment
T	The total number of time slots
n_i	Number of environments of device i
Z	Number of environments for all the devices
M	Number of flying decisions of the UAV

II. System Model and Problem Formulation

A. AoI Model for IoT Devices

Consider a geographical area in which a set $\mathcal{N}$ of N IoT devices are deployed as in Fig. 1. These devices are randomly distributed in this area according to a uniform distribution. The IoT devices are equipped with radio transceivers, microcontrollers, and sensors, enabling them to monitor and interact with their physical systems and surrounding environments [4]. For example, IoT devices in a smart factory can be deployed to monitor the production line, such as the equipments, products, etc. and their environmental conditions. Upon capturing the data, the IoT devices employ processing and data analysis to extract valuable insights, such as the equipment malfunctions, product quality, or any other useful events. In response to these detected events, each IoT device is required to take actions (e.g., triggers an alert, adjusts the manufacturing parameters, etc.). These actions are time-sensitive and require real-time responses, which necessitates striking a balance between time criticality and energy consumption of IoT devices to maintain operational efficiency and sustainability.

The coordinates of each device i are denoted as $\boldsymbol{d}_i = (x_i, y_i)$, where $i = 1, \ldots, N$. Furthermore, the system operates over a time horizon that is divided into equal time slots t, where $t = 1, 2, \ldots, T$, where T is the total number of time slots. At the beginning of each time slot t, device i collects data from its environment. Here, the environment refers to the physical system and its surroundings. We assume that the local environments of the IoT devices are independent from each other. The data packets that arrive at each device are independent and identically distributed (i.i.d.). Assume that the arrivals of data packets at device i follow a Bernoulli distribution with a probability $\lambda_{i,t}$ at time slot t. Let $q_{i,t} \in \{0, 1\}$ represent the arrival of a data packet of device i at time slot t, where $q_{i,t} = 1$ indicates the arrival of a data packet to device i; and $q_{i,t} = 0$, otherwise. Also, assume that the size $a_{i,t} \geq 0$ of a data packet follows a Gaussian distribution with parameters $(\bar{a}_{i,t}, \sigma^2_{i,t})$, where $\bar{a}_{i,t}$ is the average number of CPU cycles required to process a packet, and $\sigma_{i,t}$ is the standard deviation [1].

At each time slot t, each device i allocates a certain number of CPU cycles $\epsilon_{i,t} \in [0, \epsilon_{i,\max}]$ for processing the received packets, where $\epsilon_{i,\max}$ is the maximum number of CPU cycles of device i per time slot. Given the constant length of the timeslots, the energy consumption per timeslot is simplified to become directly proportional to $\kappa_i \epsilon^3_{i,t}$ [31], where κ_i is a paramater related to the chip architecture of the CPU. A first-come-first-serve (FCFS) policy is employed. At the end of slot t, the number of CPU cycles required to process the remaining packets in the queue of device i is given by:

$$b_{i,t+1} = \max\{b_{i,t} + q_{i,t} a_{i,t} - \epsilon_{i,t}, 0\}, \tag{1}$$

where $b_{i,t}$ is the total amount of data in the queue.

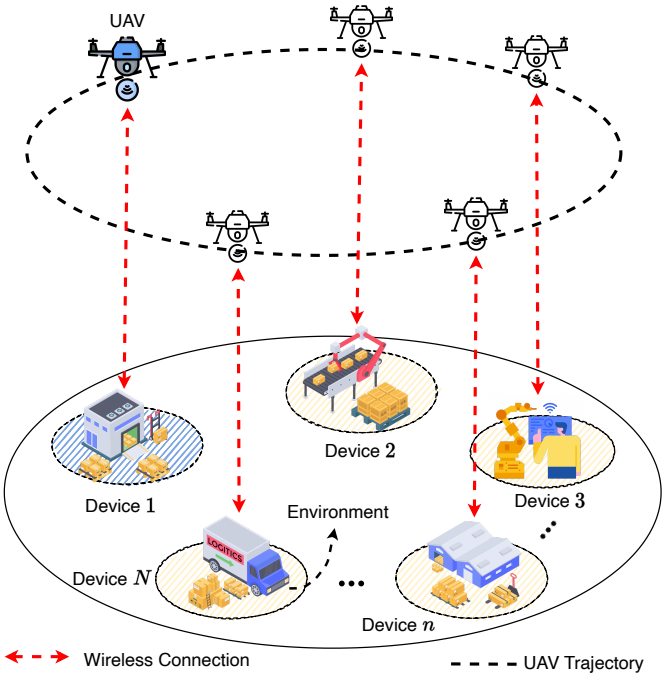

Fig. 1: An illustration of the system model comprising of a UAV that sequentially visits the IoT devices which are distributed in a real-world non-stationary environment.

To measure the freshness of information in the data packets, we consider the AoI at each device i is $\Delta_{i,t} = t - u_{\xi_{i,t}}$ during time slot t, where $\xi_{i,t}$ is the index of the most recently processed data packet at device i in time slot t and $u_{\xi_{i,t}}$ is the time at which the most recent data packet at device i was generated [5]. Then, the evolution of the AoI for each device i is stated as:

$$\Delta_{i,t+1} = \begin{cases} \Delta_{i,t} + 1\,, & \text{if} \quad \omega_{i,t+1} = 0\,, \\ (t+1) - u_{\xi_{i,t+1}}\,, & \text{if} \quad \omega_{i,t+1} = 1\,, \end{cases} \tag{2}$$

where $\omega_{i,t+1} = 1$ if the number of CPU cycles required to process the data packets at time t is sufficient (i.e., the queue is empty) to complete computing; $\omega_{i,t+1} = 0$, otherwise.

To further elucidate this concept, Fig. 2 illustrates the evolution of the AoI over time. Here, each device collects and packetizes data from its environment instantly. Upon its arrival at the IoT device, a data packet is processed immediately if there are no other packets in the queue. If there are packets ahead in the queue, the newly arrived packet incurs a queuing delay. This causes the AoI to increase until the newly arrived packet is processed, resulting in the sawtooth shape in Fig. 2. Clearly, there exists a tradeoff between the AoI and energy consumption. In other words, the more CPU cycles are allocated, the more data packets can be processed. Thus, this leads to better AoI performance, i.e., lower AoI. Henceforth, we define a cost function for each device i to capture this tradeoff, as follows:

$$c_i(t) = \beta \Delta_{i,t} + (1-\beta)\kappa_i \epsilon_{i,t}^3, \tag{3}$$

where $\beta \in [0,1]$ is a factor to balance the tradeoff between the AoI and energy consumption during each time slot t.

B. Non-Stationary Environment Model

As a result to the dynamic changes over time, the environment experienced by each IoT device varies in a non-stationary fashion. These variations are reflected in the probability of data packet arrivals and the distribution of the sizes of the arrived data packets. Thus, each IoT device faces challenges in adapting its CPU cycles allocation accordingly to its new environment. This entails dynamically adjusting the allocation strategies to meet the varying requirements and optimize overall system performance.

Definition 1. An *environment* corresponds to a unique set of environmental parameters. The environment of device i at time slot t is represented by $\boldsymbol{z}_{i,t} = (\lambda_{i,t}, \bar{a}_{i,t}, \sigma^2_{i,t}, \kappa_i, \epsilon_{i,\max})$.

In the environment tuple, κ_i and $\epsilon_{i,\max}$ remain constant for each device but may vary across different devices. These parameters represent the physical properties of each device that are inherent to the device itself and cannot be altered by the surrounding environments. However, since κ_i and $\epsilon_{i,\max}$ impact the CPU allocation strategies, it is reasonable to include them as part of the environment-specific tuple.

Furthermore, we assume that each IoT device i experiences n_i environments within the time interval $[0, T)$. Thus, $Z = \sum_{i=1}^{N} n_i$ is the total number of environments experienced by all the devices. However, it is worth noting that the value of n_i and, consequently, Z are unknown in advance. We assume that the intra-environment variations are stable over time while the inter-environment is non-stationary. The abrupt changes of the environments occur at the beginning of each time slot. The average duration of an environment for device i is p_i, which follows a Gaussian distribution with average $\bar{p}_i$ and variation σ_i'. Indeed, determining the exact distribution of p_i in advance can be challenging. In addition, the average duration of the environments remains consistent for each device, since the environments encountered by a single device follow a certain pattern or regulation.

C. UAV Model

A UAV serves as a flying BS for IoT devices. The proposed approach involves the UAV flying from one device to another to collect environment-related data. This data collection process enables the UAV to assist the devices in adapting to the dynamic environments it encounters.

Initially, the UAV starts at a known location $\boldsymbol{l}_0$. We consider $m = 1, \ldots, M$ to indicate the m-th flight made by the UAV, where M is the total number of flights that the UAV can make during T time slots. Then, we define $\boldsymbol{l}_m \in \{\boldsymbol{d}_i\}$, $\forall m \in \mathcal{M} \triangleq \{1, \ldots, M\}$. However, the exact value of M is considered to be unknown. The UAV is assumed to maintain a constant velocity $v_{\min} \leq v_m \leq v_{\max}$ when flying from one device to another, where $v_{\min}$ and $v_{\max}$ are the minimum and maximum velocities of the UAV, respectively. In addition, once the UAV arrives at a device, it will hover over it before it flies to the next destination. The distance between UAV and the device is considered close enough such that the energy consumed for their communication can be neglected. Moreover, we assume that the velocity of the UAV permits it to visit each environment for multiple times.

When flying from one location $\boldsymbol{l}_m$ to another location $\boldsymbol{l}_{m+1}$, the UAV consumes energy for propulsion, communication, and computing. Here, assume a rotary wing UAV whose total energy e_m^U is dominated by propulsion [32]. The energy of the UAV can be expressed as [32]:

$$e_0(v_m) = \underbrace{P_0\left(1+\frac{3v_m^2}{v_{\text{tip}}^2}\right)}_{\text{blade profile}} + \underbrace{P_i\left(\sqrt{1+\frac{v_m^4}{4v_0^2}}-\frac{v_m^2}{2v_0^2}\right)^{1/2}}_{\text{induced}} + \underbrace{\frac{1}{2}d_0\rho s A v_m^3}_{\text{parasite}}, \tag{4}$$

where P_0 and P_i denote the blade profile power and induced power, respectively, v_{tip} is the tip speed of the rotor blade, v_0 is the mean rotor-induced velocity in hover, d_0 is the fuselage drag ratio, s is the rotor solidity, ρ is the air density, and A is the rotor disc area. Hence, the total energy consumed by the UAV during its m-th visit can be formulated as:

$$e_m^U(v_m, \boldsymbol{l}_m, \boldsymbol{l}_{m+1}) = \frac{\|\boldsymbol{l}_{m+1} - \boldsymbol{l}_m\|}{v_m} e_0(v_m)\,. \tag{5}$$

Here, we consider two types of transmissions: a) uplink data collection and b) downlink strategy update. However, as a small amount of sampled data is typically uploaded, we can neglect the time and energy required for uplink data transmission [33]. Similarly, as the downlink transmission involves the transmission of strategies for IoT devices, it is reasonable to disregard this time and energy as well [34]. This is due to the fact that the strategies of all the devices have the same structure. In addition, we consider that the processing time and energy consumption of the UAV are mainly governed by the strategy update procedure and hovering, respectively. Based on the upcoming display of the strategy update process, it is clear that each strategy update involves similar steps and consumes the same amount of energy. Hence, it is valid to consider it as a constant within in each flight. Leveraging the fact that the strategy update procedure is similar and the distance between the UAV and the device is near, we assume the hovering time is the same. Since the hovering energy consumed by the UAV is proportional to the number of time slots it hovers [32], the hovering time and hovering energy can be treated as constant values. As this constant on each flight would not impact the optimization process, it can be ignored in the optimization. Therefore, the total energy consumed by the UAV during each flight is equivalent to e_m^U.

D. Problem Formulation

With the modeling of the UAV and IoT devices established, the next step is to formulate our objective function. Our goal is to minimize the cost for all devices and the energy consumption of the UAV. The problem can be formulated as:

$$\min_{\boldsymbol{E},\boldsymbol{v},\boldsymbol{F}} \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\mathbb{E}_{\lambda_{i,t},a_{i,t}}[c_i(t)] + \frac{\mu}{M}\sum_{m=0}^{M} e_m^U(v_m, \boldsymbol{l}_m, \boldsymbol{l}_{m+1}) \tag{6}$$

$$\text{s.t.} \quad \Delta_{i,t} \in \mathbb{N}, \quad \forall i \in \mathbb{N},\ t \in [0,T), \tag{7}$$

$$\epsilon_{i,t} \le \epsilon_{i,\max}, \forall i \in \mathbb{N},\ t \in [0,T), \tag{8}$$

$$v_{\min} \le v_m \le v_{\max},\ \forall\, m = 0,1,\ldots,M, \tag{9}$$

$$\boldsymbol{l}_m \in \{\boldsymbol{d}_i\},\ \ \forall m = 1,\ldots,M,\ \forall i = 1,\ldots,N, \tag{10}$$

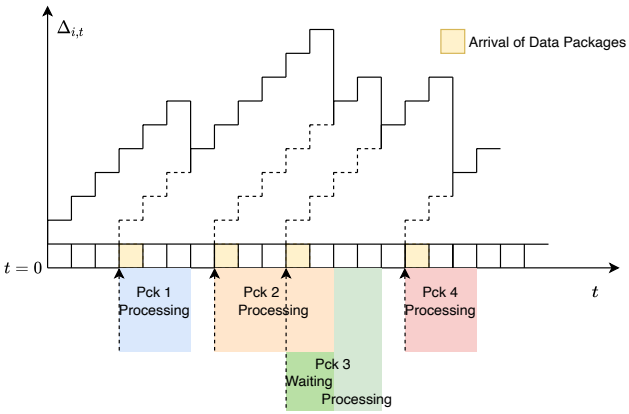

Fig. 2: The dynamic evolution of AoI for device i.

where $\mu \in [0,1]$ is a parameter that regulates the tradeoff between the devices' cost and the UAV energy usage. In addition, $\boldsymbol{\varepsilon}_i = [\epsilon_{i,0},\ldots,\epsilon_{i,T}]$ indicates the vector of CPU cycles of device i throughout the period T, $\boldsymbol{E} = [\boldsymbol{\varepsilon};\ldots;\boldsymbol{\varepsilon}_N]$ is the matrix of CPU cycles for all devices, $\boldsymbol{v} = [v_1,\ldots,v_M]$ is the vector of UAV's velocity during its flight, $\boldsymbol{F} = [\boldsymbol{l}_1,\ldots,\boldsymbol{l}_M]$ is vector of the UAV's destination during its flight. Constraints (8) and (9) indicate the limits of the CPU cycles of each device and the UAV, respectively. Constraint (10) specifies that the target point of a UAV flight is the location of the selected device. The UAV flies in a straight line to the target point. It is worth noting here that the distribution of the data packet arrival $\lambda_{i,t}$ and the distribution of the sizes of the arrived packet $a_{i,t}$ for each device i in time slot t are unknown. In addition, the duration of any environment follows an unknown distribution.

Furthermore, the UAV assists the devices in a sequential manner. When the UAV arrives at an IoT device during its m-th flight, it collects the environment related information from the device and updates its strategy. This strategy reflects the device's interaction with its current environment. In particular, it helps the device determine CPU cycle allocation at each time slot before the UAV comes for its next visit. This can help the device improve its decisions in its current environment.

To help the devices update their interacting strategies, the UAV also learns its own trajectory. It determines its flight path $\boldsymbol{F}$ and $\boldsymbol{v}$. On the one hand, the decisions made by the UAV on the flight path $\boldsymbol{F}$ and velocity $\boldsymbol{v}$ can have a strong impact on the first term in the objective (6). In essence, early or late visits of the UAV at a device can affect the cost of the device, particularly in the face of unknown changes in the environment. For instance, we can consider the scenario of a delayed visit. If the UAV visits a device significantly late after the device has already encountered a new environment, the device may acquaint an outdated and unsuitable interacting strategy in the absence of the UAV. Due to changes in data packet arrival probability and size, this inappropriate strategy can result in elevated AoI and unnecessary CPU energy consumption. Consequently, the arrival time, i.e., the flight path of the UAV, can have a vital impact on the first term in (6). On the other hand, the decision making process of the devices, i.e., $\boldsymbol{E}$, also affects the UAV's energy consumption,

i.e., the second term in (6). For instance, if a device's CPU decisions are suboptimal, i.e., its interaction strategy is poor, this can lead to repetitive UAV visits for collecting data or extended the hovering times over the device for data collection. Consequently, the probability that the UAV visits other devices with new environments decreases. This, in turn, can degrade the overall performance of the entire system. Hence, in either case, the energy consumption of the UAV increases.

Furthermore, the flight destination of the UAV $\boldsymbol{l}_m$ in (6) is discrete. Hence, the problem is a Mixed Integer Programming (MIP) problem, which is typically NP-hard and finding an optimal solution for the problem can be highly challenging. To efficiently solve this problem, we decouple it into two sub-problems that are solved in an alternating manner. First, we fix the flight control of the UAV and optimize the decision-making processes of the IoT devices. We resort to a lifelong learning solution as the environments of these devices are considerably non-stationary. The obtained optimal decisions are denoted by $\boldsymbol{E}^*$, consisting of optimal CPU cycles decision $\epsilon^*_{i,t}$ for all devices at all time slots. Subsequently, we utilize the optimal interaction strategies obtained in the first stage to calculate the UAV's optimal flying decisions using an AC network. The obtained optimal flying decisions are denoted as $\boldsymbol{F}^* = [\boldsymbol{l}^*_1, \ldots, \boldsymbol{l}^*_M]$ and $\boldsymbol{v}^* = [v^*_1, \ldots, v^*_M]$.

III. Lifelong RL for IoT Strategy Optimization in Non-stationary Environments

As discussed earlier, the two terms in the objective function (6) are optimized alternately while the other term remains fixed. In this section, we focus on the optimization of the first term in (6). As such, the flight path and flying velocities of the UAV are considered to be constant throughout the time period T. Hence, the optimization problem (6) is reduced to the following AoI-energy cost minimization problem via optimizing $\boldsymbol{E}$, that can be formulated as follows:

$$\min_{\boldsymbol{E}} \quad \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\mathbb{E}_{\lambda_{i,t},a_{i,t}}[c_i(t)] \tag{11}$$
$$\text{s.t.} \quad (7), (8).$$

During each flight, the UAV destination is randomly selected from the pool of all IoT devices. The problem in (11) involves optimizing the performance of IoT devices in an environment that undergoes discrete changes in an unpredictable manner. In other words, each environment persists for an unknown duration before being replaced by a new one. Furthermore, each environment can be considered as a stationary environment characterized by the tuple $z_{i,t}$ throughout its duration. Consequently, the entire problem described in (11) can be viewed as a collection of independent environments that appear sequentially.

Here, traditional optimization approaches, such as convex and stochastic optimization [35], are unsuitable to solve this problem, as they rely on stationary environments in which the distribution of the generated data is known. However, in our case, the dynamical patterns of the environments are unknown and non-stationary. Moreover, classical machine learning algorithms, such as supervised learning and RL, are also unsuitable for solving this problem as the setting does not take place in a stationary condition. In addition, a continuous knowledge transfer method is needed to leverage the acquired knowledge from one environment to the other. Hence, a continual learning method that allows a sequential stream of tasks to be acquired should be considered. Notably, lifelong learning has the potential to address the challenges presented by this problem. Despite the early proposal of lifelong learning in the literature [36], [37], most works have been limited to supervised learning methods and have not exploited the RL regime. Unlike these existing studies, we propose an efficient lifelong RL approach where each environment can be modeled as an RL problem and knowledge can be transferred between them accordingly.

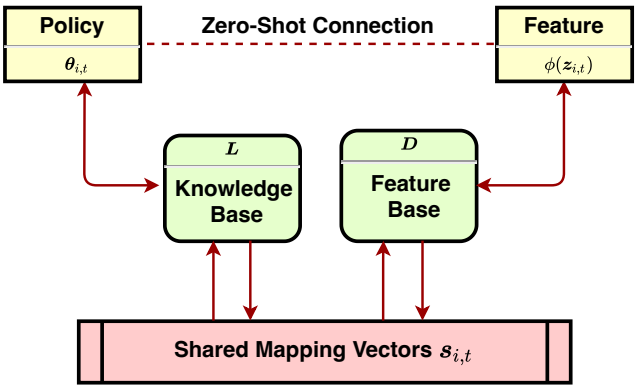

Fig. 3: An illustration of the relationship between the knowledge base and environment-related vectors.

A. RL for Independent Environments

With the independent IoT environments, as discussed earlier, we first present the optimization model of each independent environment. Subsequently, we provide its corresponding MDP. Each independent environment can be modeled through the following problem:

$$\min_{\boldsymbol{E}} \quad \mathbb{E}_{\lambda_{i,t},a_{i,t}}[c_i(t)] \tag{12}$$
$$\text{s.t.} \quad (7), (8).$$

It is worth noting that the start time and end times of each environment in time slot t are $t^s_{i,t}$ and $t^e_{i,t}$, respectively. In addition, we note that $0 \leq t^s_{i,t} < t^e_{i,t} \leq T$ while having these variables unknown in advance.

Let χ_i refer to whether device i experiences an environmental change, by having $\chi_i = 1$ indicating a new environment and $\chi_i = 0$ indicating that there is no change in the environment. For each environment, we can model the decision-making process of each device as a MDP, as delineated in the following.

In this system, each IoT device interacts with its surrounding environment at each time slot by processing data packets received from the environment. During this interaction, the device generates a record of its status changes, i.e., *interaction history*, denoted by $\boldsymbol{\tau}$. The space in which this interaction history occurs is defined as $\{\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{R}\}$, where $\mathcal{X}$ is the state

space, $\mathcal{Y}$ is the action space, and $\mathcal{R}$ is the reward function, and are elucidated as follows:

- The state space $\mathcal{X}$ is a set of tuples $\boldsymbol{x}_{i,t} = (\Delta_{i,t}, b_{i,t})$ representing the AoI and the number of CPU cycles required pending data packets at the beginning of slot t for device i. The state space has a dimension of $d = 2$, which is the number of variables in the state space.
- The action space is the set of all possible CPU cycles $\mathcal{Y} = \{\epsilon_{i,t}, \forall i, t\}$.
- The reward function $\mathcal{R}(\boldsymbol{x}_{i,t}, \epsilon_{i,t})$ is given by

$$\mathcal{R}(\boldsymbol{x}_{i,t}, \epsilon_{i,t}) = -(\beta\Delta_{i,t} + (1-\beta)\kappa_i\epsilon_{i,t}^3).$$

- $\Pi_{i,t}$ is defined as the set of policy parameters:

$$\Pi_{i,t} = \{\pi_{\boldsymbol{\theta}_{i,t}} | \boldsymbol{\theta}_{i,t} \in \mathbb{R}^d\},$$

where $\boldsymbol{\theta}_{i,t}$ is the policy for device i at time slot t. In addition, $\pi_{\boldsymbol{\theta}_{i,t}}(\epsilon_{i,t}|\boldsymbol{x}_{i,t}) = \Pr\{\epsilon_{i,t}|\boldsymbol{x}_{i,t}, \boldsymbol{\theta}_{i,t}\}$ is the parameterized function that determines the probability of selecting a particular action $\epsilon_{i,t}$, given state $\boldsymbol{x}_{i,t}$. The goal is to find the optimal decision-making policy $\pi_{\boldsymbol{\theta}^*_{i,t}}$ that minimizes the cost function for device i in the current environment.

Furthermore, each environmental MDP episode can be equivalently represented through a one-to-one mapping utilizing the variables of $\boldsymbol{z}_{i,t}$. Subsequently, we use an environment feature vector $\phi(\boldsymbol{z}_{i,t}) \in \mathbb{R}^{d_z}$ to represent each unique MDP episode. $\phi(\cdot)$ is a feature extraction function, and d_z is its dimension. While multiple features can describe the same environment, different environments typically have unique features. However, the exact environment descriptor $\boldsymbol{z}_{i,t}$ is unknown and must be estimated from the collected interaction history $\boldsymbol{\tau}$.

Up so far, we have acknowledged the independence between environments. We proceed to determine the environment change detection. It is eligible to consider that the change of $\boldsymbol{z}_{i,t}$ indicates the change of environments. As such, $\boldsymbol{z}_{i,t}$ can be detected by sampling and collecting the environment related information. The environment related information can be extracted from interaction history $\boldsymbol{\tau}$. Given interaction history $\boldsymbol{\tau} = \{\boldsymbol{x}_{i,t}, \epsilon_{i,t}, \mathcal{R}(\boldsymbol{x}_{i,t}, \epsilon_{i,t})\}$, where $t^s_{i,t} \le t \le t^e_{i,t}$. Then $(t^e_{i,t} - t^s_{i,t})$ is the length of the interaction history and $\mathcal{Q}_i = \{t^s_{i,t} \le t \le t^e_{i,t}, \, | \, q_{i,t} = 1, t \in \mathbb{N}\}$ is the set of time slots when there was a packet arrival. The environment descriptor can be estimated as follows: $\tilde{\lambda}_{i,t} \approx |\mathcal{Q}_i|/(t^e_{i,t} - t^s_{i,t})$, $\tilde{a}_{i,t} \approx \sum_{t\in\mathcal{Q}_i}(b_{i,t+1} - b_{i,t})/|\mathcal{Q}_i|$, and $\tilde{\sigma}^2_{i,t} \approx [\sum_{t\in\mathcal{Q}_i}(b_{i,t+1} - b_{i,t} - \bar{a}_{i,t})]/|\mathcal{Q}_i|$. Here, the device-specific parameters κ_i and $\epsilon_{i,\max}$ can be directly obtained from the IoT device. By combining the estimated values of $\tilde{\lambda}_{i,t}$, $\tilde{a}_{i,t}$ and $\tilde{\sigma}^2_{i,t}$, the environment descriptor $\tilde{\boldsymbol{z}}_{i,t}$ can be obtained. This process is dubbed as *environment discovery*.

Up until now, there are two variables that capture these high-level features for each single environment: i) the environment policy $\boldsymbol{\theta}_{i,t}$ and ii) the environment feature vector $\phi(\boldsymbol{z}_{i,t})$. Here, $\boldsymbol{\theta}_{i,t}$ determines the cost of each device at any time slot, while $\phi(\boldsymbol{z}_{i,t})$ identifies a specific environment. Both variables are independently distributed, as the environment distribution is i.i.d and unknown. Thus, $\boldsymbol{\theta}_{i,t}$ and $\phi(\boldsymbol{z}_{i,t})$ contain different attributes of an environment [38]. Hence, it is worth exploiting the similarities between these attributes to share experiences across various environments.

Algorithm 1 UpdateL

if $\chi_i = 0$ **then**
  $\boldsymbol{A_L} \leftarrow \boldsymbol{A_L} - \left(\boldsymbol{s}\boldsymbol{s}^{\mathrm{T}}\right) \otimes \boldsymbol{\Gamma}$
  $\boldsymbol{b_L} \leftarrow \boldsymbol{b_L} - \mathrm{vec}\left(\boldsymbol{s}^{\mathrm{T}} \otimes \left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Gamma}\right)\right)$
else
  Identify a new environment for device i
  $Z \leftarrow Z + 1$
end if
$\boldsymbol{A_L} \leftarrow \boldsymbol{A_L} + \left(\boldsymbol{s}\boldsymbol{s}^{\mathrm{T}}\right) \otimes \boldsymbol{\Gamma}$
$\boldsymbol{b_L} \leftarrow \boldsymbol{b_L} + \mathrm{vec}\left(\boldsymbol{s}^{\mathrm{T}} \otimes \left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Gamma}\right)\right)$
$\boldsymbol{L} \leftarrow \mathrm{mat}\Big(\left(\frac{1}{Z}\boldsymbol{A_L} + \eta_3 \boldsymbol{I}_{d\times h, d\times h}\right)^{-1} \frac{1}{Z}\boldsymbol{b_L}\Big)$

Algorithm 2 Lifelong Reinforcement Learning

Require: $T \leftarrow 0$, $\boldsymbol{A} \leftarrow \mathrm{zeros}_{d\times h, d\times h}$, $\boldsymbol{b} \leftarrow \mathrm{zeros}_{d\times h, 1}$
Require: $\boldsymbol{L} \leftarrow \mathrm{zeros}_{d,h}$, $\boldsymbol{D} \leftarrow \mathrm{zeros}_{d_z,h}$
while UAV arrives at device i **do**
  Collect $\boldsymbol{\tau}$
  Identify environment feature $\phi(\tilde{\boldsymbol{z}})$ for device i using $\boldsymbol{\tau}$
  Update χ_i according to $\phi(\tilde{\boldsymbol{z}})$
  Compute $\boldsymbol{\alpha}_{i,t}$ and $\boldsymbol{\Gamma}_{i,t}$ from $\boldsymbol{\tau}$
  $\boldsymbol{L}, \boldsymbol{D} \leftarrow \mathrm{reinitializeAllZeroColumns}(\boldsymbol{L}, \boldsymbol{D})$
  $\boldsymbol{s} \leftarrow \arg\min_{\boldsymbol{s}} \ell\left(\boldsymbol{K}, \boldsymbol{s}, \boldsymbol{\beta}, \boldsymbol{Q}\right)$
  $\boldsymbol{L} \leftarrow \mathrm{updateL}\ell(\boldsymbol{L}, \boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{\Gamma})$
  $\boldsymbol{D} \leftarrow \mathrm{updateD}\ell(\boldsymbol{D}, \boldsymbol{s}, \boldsymbol{\phi}(\tilde{\boldsymbol{z}}), \eta_1 \boldsymbol{I}_{d_z})$
end while

B. Lifelong Learning for Non-stationary Environments

Our goal is to achieve a balance between the AoI and energy consumption for all devices. Given that the devices experience independent environments and with the aforementioned MDP model, problem (11) can be rewritten as:

$$\min_{\Pi_{i,t}} \frac{1}{Z} \sum_{i=1}^{N} \sum_{t=0}^{T} \mathcal{J}(\boldsymbol{\theta}_{i,t}), \tag{13}$$

where $\mathcal{J}(\boldsymbol{\theta}_{i,t}) = \int p_{\boldsymbol{\theta}_{i,t}}(\boldsymbol{\tau}) \mathfrak{R}_{\mathrm{i,t}}(\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\tau}$, $p_{\boldsymbol{\theta}_{i,t}}$ is the probability distribution of interaction history $\boldsymbol{\tau}$, and $\mathfrak{R}_{\mathrm{i,t}}(\boldsymbol{\tau})$ is the reward of the trajectory τ. As such, we can formulate the aforementioned variables as:

$$p_{\boldsymbol{\theta}_{i,t}}(\boldsymbol{\tau}) = P_0(\boldsymbol{x}_0) \prod_{t=t^s_{i,t}}^{t^e_{i,t}} p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t}, \epsilon_{i,t}) \, \pi_{\boldsymbol{\theta}_{i,t}}(\epsilon_{i,t}|\boldsymbol{x}_{i,t}), \tag{14}$$

$$\mathfrak{R}_{\mathrm{i,t}}(\boldsymbol{\tau}) = \frac{1}{t^s_{i,t} - t^e_{i,t}} \sum_{t=t^s_{i,t}}^{t^e_{i,t}} \mathcal{R}(\boldsymbol{x}_{i,t}, \epsilon_{i,t}), \tag{15}$$

where $p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t}, \epsilon_{i,t})$ is the unknown state transition probability that maps a state-action pair at time slot t onto a distribution of states at time slot $t+1$.

The problem in (13) involves a sequential stream of independent reinforcement environments as denoted in III-A.

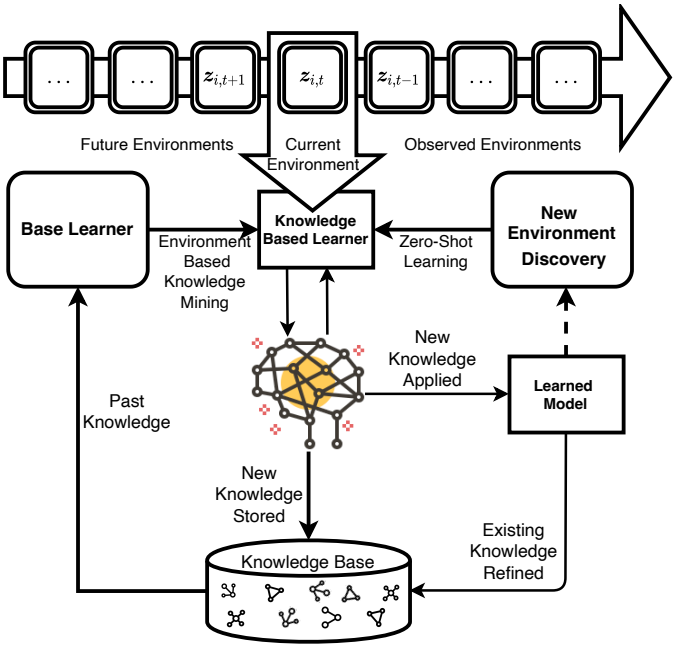

Fig. 4: The flow diagram of the proposed lifelong RL algorithm.

To enable continuous learning throughout the dynamical RL environments, it is important to explore and exploit the common structures revealed by the high level features such as $\boldsymbol{\theta}_{i,t}$ and $\phi(\boldsymbol{z}_{i,t})$. To enable knowledge transfer between environments, we assume that the policy $\boldsymbol{\theta}_{i,t}$ is a linear combination of h latent components [28], i.e., $\boldsymbol{\theta}_{i,t} = \boldsymbol{L}\boldsymbol{s}_{i,t}$, where $\boldsymbol{s}_{i,t} \in \mathbb{R}^h$ is a vector of linear parameters and $\boldsymbol{L}$ is a knowledge base with a library of h latent components that represents the shared knowledge of all the environments. The dimension of h, denoted as d, is chosen independently with cross-validation. The mapping function $\boldsymbol{s}_{i,t}$ should be sparse to maximize the knowledge captured by the latent components. As such, each observed environment can be a linear combination of only a few latent components in $\boldsymbol{L}$. To incorporate the environment feature, we also assume that the environment feature vector can be linearly represented by a latent basis $\boldsymbol{D} \in \mathbb{R}^{d_z \times h}$. Similar to the knowledge base $\boldsymbol{L}$, the environment feature basis $\boldsymbol{D}$ can capture the commonalities among the environment descriptors, such as $\phi(\boldsymbol{z}_{i,t}) = \boldsymbol{D}\boldsymbol{s}_{i,t}$. As illustrated in Fig. 3, the policy base $\boldsymbol{L}$ and feature base $\boldsymbol{D}$ share the same coefficient vectors $\boldsymbol{s}_{i,t}$. As such, the environment policy $\boldsymbol{\theta}_{i,t}$ and the environment feature vector $\phi(\boldsymbol{z}_{i,t})$ can be connected through the shared mapping vectors. It is reasonable to utilize the relationship between the feature dictionary and the knowledge base.

We proceed to optimize the coupled bases $\boldsymbol{L}$ and $\boldsymbol{D}$ together. Up to this end, techniques from the field of sparse coding are utilized. Specifically, coupled dictionary optimization [29] is applied to optimize the dictionaries for multiple feature spaces with a joint sparse representation. The result of incorporating knowledge transfer and feature coding into the optimization process is a multi-environment loss function based on coupled dictionaries, given as:

$$g_{\mathrm{T}}(\boldsymbol{L},\boldsymbol{D}) = \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\min_{\boldsymbol{s}_{i,t}}\Big[\mathcal{J}(\boldsymbol{\theta}_{i,t}) + \eta_1\|\phi(\boldsymbol{z}_{i,t}) - \boldsymbol{D}\boldsymbol{s}_{i,t}\|_2^2 + \eta_2\|\boldsymbol{s}_{i,t}\|_1\Big] + \eta_3(\|\boldsymbol{L}\|_{\mathrm{F}}^2 + \|\boldsymbol{D}\|_{\mathrm{F}}^2)\,, \tag{16}$$

where ℓ_1-norm approximates the vector sparsity and $\|\boldsymbol{L}\|_{\mathrm{F}} = (tr(\boldsymbol{L}\boldsymbol{L}'))^{1/2}$ is the Frobenius norm of matrix $\boldsymbol{L}$. The parameter η_1 controls the balance between the policy's fit and the feature's fit. Also, η_2 and η_3 are two regularization parameters, where η_2 controls the sparsity of $\boldsymbol{s}_{i,t}$. The penalty on the Frobenius norm of $\boldsymbol{L}$ and $\boldsymbol{D}$ regularizes the predictor weights to have low ℓ_2-norm and avoids overfitting.

The optimal $\boldsymbol{\theta}_{i,t}$ in (16) can achieve the minimum $g_{\mathrm{T}}(\boldsymbol{L},\boldsymbol{D})$, having $\boldsymbol{L}$ and $\boldsymbol{D}$ given. Considering that $\boldsymbol{\theta}_{i,t} = \boldsymbol{L}\boldsymbol{s}_{i,t}$, (16) is transformed into a minimization problem based on $\{\boldsymbol{s}_{i,t}\}$. Consequently, we can first obtain $\boldsymbol{L}$ and $\boldsymbol{D}$ with a series of consecutive environments. Then, $\boldsymbol{s}_{i,t}$ can be further optimized. Given $\boldsymbol{L}$ and $\boldsymbol{s}_{i,t}$, the suboptimal $\boldsymbol{\theta}_{i,t}$ can be obtained eventually.

To compute $\boldsymbol{L}$ and $\boldsymbol{D}$, we need to access the interaction history of all environments for all devices, as evident from (16). Notably, there remains dependence between the policies of environments and their interaction history. To suppress this dependence, we use a second-order Taylor expansion to approximate $\mathcal{J}(\boldsymbol{\theta}_{i,t})$ around an estimated optimal policy, denoted as $\boldsymbol{\alpha}_{i,t}$, for each individual environment. The estimated policy is defined as:

$$\boldsymbol{\alpha}_{i,t} = \arg\min_{\boldsymbol{\theta}_{i,t}} \mathcal{J}(\boldsymbol{\theta}_{i,t}). \tag{17}$$

Here, the method used to obtain $\boldsymbol{\alpha}_{i,t}$ is the base learner. Accordingly, we use the collected interaction history of device i to evaluate $\boldsymbol{\alpha}_{i,t}$.

The second-order Taylor expression of $\mathcal{J}(\boldsymbol{\theta}_{i,t})$ is expanded around $\boldsymbol{\alpha}_{i,t}$. For each environment $z_{i,t}$,

$$\mathcal{J}(\boldsymbol{\theta}_{i,t} = \boldsymbol{L}\boldsymbol{s}_{i,t}) = \mathcal{J}(\boldsymbol{\alpha}_{i,t}) + \nabla\mathcal{J}(\boldsymbol{\theta}_{i,t})_{\boldsymbol{\theta}_{i,t}=\boldsymbol{\alpha}_{i,t}}(\boldsymbol{\alpha}_{i,t} - \boldsymbol{L}\boldsymbol{s}_{i,t}) + \|\boldsymbol{\alpha}_{i,t} - \boldsymbol{L}\boldsymbol{s}_{i,t}\|_{\boldsymbol{\Gamma}_{i,t}}^2, \tag{18}$$

where $\nabla\mathcal{J}(\boldsymbol{\theta}_{i,t})$ is the first-order gradient of $\mathcal{J}(\boldsymbol{\theta}_{i,t})$ and $\boldsymbol{\Gamma}_{i,t}$ is the Hessian matrix. The first term on the right-hand side (RHS) of (18) is a constant and can be suppressed. The second term takes a negligible value as $\boldsymbol{\alpha}_{i,t}$ is the minimizer of (17). Substituting the second-order Taylor expansion into (16) yields the following loss function:

$$g_{\mathrm{T}}(\boldsymbol{L},\boldsymbol{D}) = \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\min_{\boldsymbol{s}_{i,t}}\Big[\|\boldsymbol{\alpha}_{i,t} - \boldsymbol{L}\boldsymbol{s}_{i,t}\|_{\boldsymbol{\Gamma}_{i,t}}^2 + \eta_1\|\phi(\boldsymbol{z}_{i,t}) - \boldsymbol{D}\boldsymbol{s}_{i,t}\|_2^2 + \eta_2\|\boldsymbol{s}_{i,t}\|_1\Big] + \eta_3(\|\boldsymbol{L}\|_{\mathrm{F}}^2 + \|\boldsymbol{D}\|_{\mathrm{F}}^2)\,. \tag{19}$$

Now, considering the symmetrical characteristic in (19), the following pairs can be abstracted as follows:

$$\boldsymbol{\beta}_{i,t} = \begin{bmatrix}\boldsymbol{\alpha}_{i,t}\\ \phi(\boldsymbol{z}_{i,t})\end{bmatrix};\quad \boldsymbol{K} = \begin{bmatrix}\boldsymbol{L}\\ \boldsymbol{D}\end{bmatrix};\quad \boldsymbol{Q}_{i,t} = \begin{bmatrix}\boldsymbol{\Gamma}_{i,t} & \boldsymbol{0}\\ \boldsymbol{0} & \eta_1\boldsymbol{I}_{d_z}\end{bmatrix}, \tag{20}$$

Algorithm 3 Zero-Shot Transfer for a New Task

Require: Trained $\boldsymbol{L} \in \mathbb{R}^{d\times h}$ and $\boldsymbol{D} \in \mathbb{R}^{d_z\times h}$

Estimate $\phi(\tilde{z}_{i,t})$ from collected interaction history τ

$\boldsymbol{s}^*_{i,t} \leftarrow \arg\min_{\boldsymbol{s}_{i,t}} \|\phi(\tilde{\boldsymbol{z}}_{i,t}) - \boldsymbol{D}\boldsymbol{s}_{i,t}\|_2^2 + \eta_2\|\boldsymbol{s}_{i,t}\|_1$

$\boldsymbol{\theta}^*_{i,t} = \boldsymbol{L}\boldsymbol{s}^*_{i,t}$

where $\mathbf{0}$ is the all-zero matrix. With this abstraction, (19) is simplified to the following:

$$g_{\mathrm{T}}(\boldsymbol{K}) = \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\min_{\boldsymbol{s}_{i,t}}\Big[\|\boldsymbol{\beta}_{i,t} - \boldsymbol{K}\boldsymbol{s}_{i,t}\|^2_{\boldsymbol{Q}_{i,t}} + \eta_2\|\boldsymbol{s}_{i,t}\|_1\Big] + \eta_3\|\boldsymbol{K}\|_{\mathrm{F}}^2 . \tag{21}$$

Clearly, (21) is a joint optimization problem for $\boldsymbol{\beta}_{i,t}$ and $\boldsymbol{K}$. Solving this problem requires the availability of trajectories of all the devices throughout the time period T to compute the Hessian matrices $\boldsymbol{\Gamma}_{i,t}$, i.e., $\boldsymbol{Q}_{i,t}$. However, this is not feasible in practice. Firstly, predicting future dynamics beforehand is unattainable. Secondly, the UAV is limited to accessing one device at any given moment. Hence, only the trajectory of the currently visited device by the UAV can be obtained. Thus, an approach that can eliminate these dependencies is necessary.

Next, we adopt an alternating approach to optimize the shared knowledge base and device specific knowledge. First, we leverage the initialized $\boldsymbol{K}$ to optimize $\boldsymbol{s}_{i,t}$ of the current device and the environment it is experiencing. Then, we utilize the obtained $\boldsymbol{s}_{i,t}$ to optimize $\boldsymbol{L}$ and $\boldsymbol{D}$. With that in mind, we can rewrite (21) as follows:

$$\boldsymbol{s}_{i,t} \leftarrow \arg\min_{\boldsymbol{s}_{i,t}} \ell\big(\boldsymbol{K}, \boldsymbol{s}_{i,t}, \boldsymbol{\beta}_{i,t}, \boldsymbol{Q}_{i,t}\big) , \tag{22}$$

$$\boldsymbol{K} = \arg\min_{\boldsymbol{K}} \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\ell\big(\boldsymbol{K}, \boldsymbol{s}_{i,t}, \boldsymbol{\beta}_{i,t}, \boldsymbol{Q}_{i,t}\big) + \eta_3\|\boldsymbol{K}\|_{\mathrm{F}}^2 , \tag{23}$$

where

$$\ell\big(\boldsymbol{K}, \boldsymbol{s}_{i,t}, \boldsymbol{\beta}_{i,t}, \boldsymbol{Q}_{i,t}\big) = \|\boldsymbol{\beta}_{i,t} - \boldsymbol{K}\boldsymbol{s}_{i,t}\|^2_{\boldsymbol{Q}_{i,t}} + \eta_2\|\boldsymbol{s}_{i,t}\|_1 . \tag{24}$$

By fixing $\boldsymbol{K}$, $\boldsymbol{s}_{i,t}$ can be updated as in (22). This is an ℓ_1-regularized regression problem that can be solved as an instance of Lasso.

To update $\boldsymbol{K}$, we decouple (23) between $\boldsymbol{L}$ and $\boldsymbol{D}$. Since the two variables have similar structures, their update processes are identical. Hence, we consider $\boldsymbol{L}$ as an example and define its associated loss function $\ell\big(\boldsymbol{L}, \boldsymbol{s}_{i,t}, \boldsymbol{\alpha}_{i,t}, \boldsymbol{\Gamma}_{i,t}\big)$. Accordingly, $\boldsymbol{L}$ is updated through:

$$\boldsymbol{L} = \arg\min_{\boldsymbol{L}} \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\ell\big(\boldsymbol{L}, \boldsymbol{s}_{i,t}, \boldsymbol{\alpha}_{i,t}, \boldsymbol{\Gamma}_{i,t}\big) + \eta_3\|\boldsymbol{L}\|_{\mathrm{F}}^2 . \tag{25}$$

Herein, $\boldsymbol{L}$ can be easily obtained by nulling the gradient of (25). Then $\boldsymbol{L}$ can be obtained as $\boldsymbol{A}_{\boldsymbol{L}}^{-1}\boldsymbol{b}_{\boldsymbol{L}}$, where:

$$\boldsymbol{A}_{\boldsymbol{L}} = \eta_3\boldsymbol{I}_{d\times h, d\times h} + \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\big(\boldsymbol{s}_{i,t}\boldsymbol{s}_{i,t}^{\mathrm{T}}\big)\otimes\boldsymbol{\Gamma}_{i,t}, \tag{26}$$

$$\boldsymbol{b}_{\boldsymbol{L}} = \frac{1}{Z}\sum_{i=1}^{N}\sum_{t=0}^{T}\mathrm{vec}\big(\boldsymbol{s}_{i,t}^{\mathrm{T}}\otimes\big(\boldsymbol{\alpha}_{i,t}^{\mathrm{T}}\boldsymbol{\Gamma}_{i,t}\big)\big) . \tag{27}$$

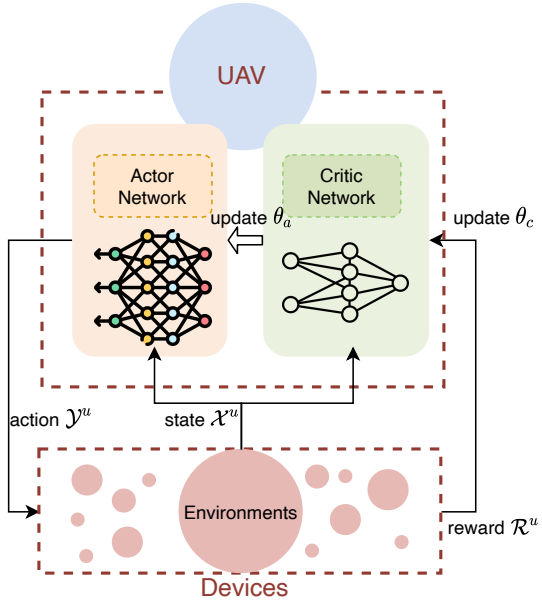

Fig. 5: An illustration of the proposed AC network for UAV flight control.

The UAV repeats the above process for each environment until $\boldsymbol{L}$ converges. This process is referred to as *updateL*, that is summarized in Algorithm 1. By assembling $\boldsymbol{s}_{i,t}$, $\boldsymbol{\alpha}_{i,t}$, and $\boldsymbol{\Gamma}$ over i and t, a similar algorithm can be used to update $\boldsymbol{D}$, which is referred to as *updateD*. Even though only a single environment is considered at a time, the policy improvement of the other environments can be obtained by improving the knowledge base $\boldsymbol{L}$ and the feature dictionary $\boldsymbol{D}$. The complete flow of this approach is presented in Algorithm 2, and a corresponding flow diagram is presented in Fig. 4 to summarize this approach. After proper training, the UAV can quickly find a high-fidelity policy $\mathcal{E}^* = \{\epsilon^*_{i,t}\}$ for each environment encountered.

C. Zero-Shot Learning Method

Thus far, we have achieved the learning and knowledge accumulation for environments with the help of $\boldsymbol{\alpha}_{i,t}$. Next, we design an algorithm that can adapt to unknown environments. To further accelerate the learning of new environments, we adopt a zero-shot transfer method [39] with coupled dictionaries [40]. The zero-shot method is often used to establish a connection between unobserved and observed classes in machine learning. Thus, it is useful in the following cases: 1) The unobserved classes are rare, and it is not easy to find adequate instances for training; 2) The total number of classes is large, and it is impossible to get all instances labeled or train all classes; 3) It is expensive to obtain instances for certain classes; 4) Target classes change over time and it is costly to label every class that is observed.

In the considered non-stationary environment, the new environments are not encountered before during during training. In other words, we do not have $\boldsymbol{\alpha}_{i,t}$ for these new environments. Hence, we leverage the zero-shot method to associate the trained environments and new environments via the shared

knowledge bases $\boldsymbol{L}$ and $\boldsymbol{D}$. Here, the feature descriptor $\phi(\boldsymbol{z}_{i,t})$ acts as a high-level descriptor of the learning process.

According to [29], a policy $\boldsymbol{\theta}_{i,t}$ in the policy parameter space can be recovered using the coupled dictionary $\boldsymbol{L}$ and $\boldsymbol{D}$ with a feature descriptor for each environment. We use the estimated value $\tilde{\boldsymbol{z}}_{i,t}$ obtained with a small set of interaction histories $\boldsymbol{\tau}$, and then obtain the shared mapping vector using the estimated feature descriptor $\phi(\tilde{\boldsymbol{z}}_{i,t})$. The loss function $\ell(\boldsymbol{D}, \boldsymbol{s}, \boldsymbol{\phi}(\tilde{\boldsymbol{z}}), \eta_1 \boldsymbol{I}_{d_z})$ in (24) can be rewritten as:

$$\boldsymbol{s}^*_{i,t} \leftarrow \arg\min_{\boldsymbol{s}_{i,t}} \{\|\phi(\tilde{\boldsymbol{z}}_{i,t}) - \boldsymbol{D}\boldsymbol{s}_{i,t}\|_2^2 + \eta_2 \|\boldsymbol{s}_{i,t}\|_1\}. \quad (28)$$

Having obtained the policy parameter $\boldsymbol{\theta}^*_{i,t}$, the optimal interacting decision $\mathcal{E}^* = \{\epsilon^*_{i,t}\}$ can be determined using a stochastic policy, i.e., $\epsilon^*_{i,t} = \boldsymbol{\theta}^*_{i,t}\boldsymbol{x}_{i,t} + \epsilon_z$, where ϵ_z is the noise of the stochastic policy having $\epsilon_z \sim \mathcal{N}(0, \sigma_z^2)$ with σ_z being the standard deviation of Gaussian distribution. This procedure is summarized in Algorithm 3.

IV. Energy Efficient UAV Trajectory and Velocity Optimization

In the previous section, we formulate the lifelong RL method for non-stationary optimization for IoT devices. However, the second term in (6) also needs to be addressed. In this section, we optimize the flight decisions $\boldsymbol{F}$ and $\boldsymbol{v}$, i.e., trajectory and velocity of the UAV, respectively. We note here that the UAV is responsible for maintaining the knowledge bases $\boldsymbol{L}$ and $\boldsymbol{D}$ for all devices and taking the corresponding flight decisions.

A. Energy Optimization of UAV

With the optimal interaction policies obtained in Section III, (6) can be reformulated. Particularly, the flight control objective of the UAV can be formulated in the form of an optimization problem as follows:

$$\min_{\boldsymbol{v}, \boldsymbol{F}} \quad \mu \frac{1}{M} \sum_{m=0}^{M} e_m^U(v_m, \boldsymbol{l}_m, \boldsymbol{l}_{m+1}) + \frac{1}{Z} \sum_{i=1}^{N} \sum_{t=0}^{T} c_i^*(t) \quad (29)$$

$$\text{s.t.} \quad (9), (10),$$

$$c_i^*(t) = \beta \Delta_{i,t} + (1-\beta)\kappa_i \epsilon^{*3}_{i,t}. \quad (30)$$

Indeed, traditional RL may not be adequate to handle the non-stationarity surrounding the IoT devices in problem (29). However, it can proficiently manage flight control of the UAV over all the devices. This is due to the fact that the non-stationarity affects the distribution of the periods, i.e., $\bar{p}_i$ and σ_i', of the IoT devices only. As such, the flight decisions of the UAV are not affected the distribution changes in the environment of the devices, but rather by the frequency of change in the underlying environment. Thereby, the UAV operates at a higher level of abstraction from the IoT devices.

Accordingly, RL methods such as value-based learning and policy-based learning can be potential solutions for UAV flight control. However, the unstable reward caused by the unpredictable environmental changes can slow down value-based learning, and the batch learning required for policy-based learning can greatly drain training resources. Hence, we

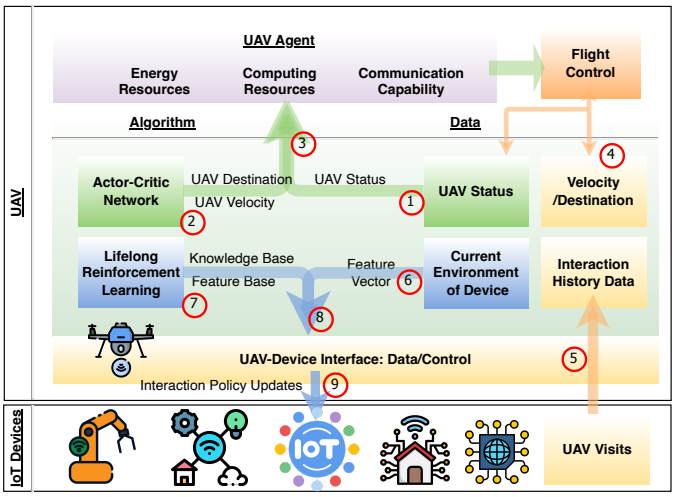

Fig. 6: Illustrative figure of the proposed lifelong RL and AC network solution for the UAV-aided non-stationary IoT network optimization.

resort to an AC framework that offers increased stability, better convergence, and reduced variations. The AC network consists of two networks: i) the actor network that is responsible for decision-making and outputs actions based on the current state inputs, and ii) the critic network, which interacts with the environment using the actions from the actor network and updates its value output accordingly. This value can be used as a judgment value in the actor network to increase or decrease the probability of the chosen action. The two networks interact with each other and the environment in an interactive manner until the optimal flight policy is obtained. An illustration of the AC network functionality is described in Fig. 5. Furthermore, we describe the denoted RL environment as follows:

- The state space of the UAV, denoted as $\mathcal{X}^u \in \mathbb{R}^{N+2}$, includes the UAV's current location and the time that has elapsed since the current environment arrived at each device. Specifically, $\mathcal{X}^u = \{\varsigma_m\} = \{\boldsymbol{l}_m, \varpi_0, \varpi_1, \ldots, \varpi_N\}$. For the device that the UAV is visiting, ϖ_i is initialized based on the duration of the current environment when the current environment is new. If the environment is not new, ϖ_i is updated by adding the flying time of the UAV to the current device. Additionally, ϖ_i is augmented by the flying time for other devices.
- The action space of the UAV, denoted by $\mathcal{Y}^u$, is defined as its next destination $\boldsymbol{l}_{m+1}$ and velocity $\boldsymbol{v}_m$, i.e., $\mathcal{Y}^u = \{\zeta_m\} = \{v_m, \boldsymbol{l}_{m+1} | v_m \leq v_{\max}, \boldsymbol{l}_{m+1} \in \mathbb{R}^2\}$, since the UAV only accesses one device at a time.
- The reward of the UAV, denoted by $R^u(\varsigma_m, \zeta m)$, is a function of the state and action; i.e., $R^u(\varsigma_m, \zeta_m) = \mu e_m^U(v_m, \boldsymbol{l}_m, \boldsymbol{l}_{m+1}) + \frac{1}{N}\sum_{i=1}^{N}\sum_{t_m}^{t=t_{m+1}} c_i(t)$. Here, each time the UAV selects a device to visit, it only considers the reward of that specific device in the current flight period.

We use $\Pi_a^u = \{\pi_{\boldsymbol{\theta}_a}\}$ to denote the distribution of the actions over the states, and $\Pi_c^u = \{\pi_{\boldsymbol{\theta}_c}\}$ to denote the action-value function parameter, where $\pi_{\boldsymbol{\theta}_a}(\zeta_m | \varsigma_m) = \Pr\{\zeta_m | \varsigma_m, \boldsymbol{\theta}_a\}$. The AC network maximizes the accumulated discounted re-

ward $G_m = -\sum_{k=m}^{\infty} \eta_4^{k-m} R^u(\varsigma_k, \boldsymbol{\zeta}_k)$, where $\eta_4 \in (0, 1]$ is the discount coefficient. Given a state and the policy, the gain of an action is evaluated by an action-value function, i.e., the Q-function, that is denoted as:

$$Q_{\pi_c}(\varsigma_m, \boldsymbol{\zeta}_m) = \mathbb{E}_{\boldsymbol{\theta}_c}[G_m | \varsigma_m, \boldsymbol{\zeta}_m]\,. \tag{31}$$

Furthermore, the Bellman expectation equation of the action-value function is given as:

$$\begin{aligned} Q_{\pi_c}(\varsigma_m, \boldsymbol{\zeta}_m) =& \mathbb{E}_{R, \varsigma \sim \Xi}\big[R^u(\varsigma_m, \boldsymbol{\zeta}_m) \\ &+ \eta_4 \mathbb{E}_{\boldsymbol{\zeta}_{m+1} \sim \pi_a}[Q_{\pi_c}(\varsigma_{m+1}, \boldsymbol{\zeta}_{m+1})]\big]\,, \end{aligned} \tag{32}$$

where Ξ is the non-statioinary environment that the devices interact with. Thus, the actor network adjusts its policy based on the action-value function that is formulated as:

$$\bigtriangledown_{\boldsymbol{\theta}_a} J(\boldsymbol{\theta}_a) = \mathbb{E}\Bigg[\sum_{m=0}^{\infty} \bigtriangledown_{\boldsymbol{\theta}_a} \log \pi_{\boldsymbol{\theta}_a}(\boldsymbol{\zeta}_{\mathrm{m}} | \varsigma_{\mathrm{m}}) \mathrm{Q}_{\pi^{\mathrm{c}}}(\varsigma_{\mathrm{m}}, \boldsymbol{\zeta}_{\mathrm{m}})\Bigg]\,. \tag{33}$$

Then, the parameter of the actor network is updated through the following equation:

$$\boldsymbol{\theta}_a \leftarrow \boldsymbol{\theta}_a + \eta_a \bigtriangledown_{\boldsymbol{\theta}_a} J(\boldsymbol{\theta}_a)\,, \tag{34}$$

where η_a is the learning rate of the AC network. Hence, the network allows the UAV to effectively learn its trajectory on the fly.

B. Algorithm Overview

In Algorithm 4, we provide detailed procedures for the training processes of the proposed lifelong RL solution with AC network. As a complement, a complete workflow of the testing procedures is illustrated in Fig. 6. Additional information regarding the training and testing procedures can be found in the simulation section. As shown in Fig. 6, ① the UAV detects and obtains its current status. ② The UAV decides its next flying destination and velocity accoring to the AC network. ③ The UAV uses its central computing resources and energy to control the flight to the destination. ④ The UAV arrives at the destination and updates its status. ⑤ When the UAV arrives at the device, it collects interaction history data from the device. ⑥ The UAV obtains the feature vector of the current environment. ⑦ The proposed Lifelong RL method is used to compute the policy for the current device. ⑧ The UAV transmits the environment based policy to the device through interface with the device. ⑨ The devices interact with the surrounding environments using the received policies.

C. Complexity Analysis

For the lifelong reinforcement learning algorithm, each update step begins with a UAV's visit to a device. The update process begins by obtaining $\boldsymbol{\alpha}_{i,t}$ and $\boldsymbol{\Gamma}_{i,t}$ for the encountered environment. We employ a base learner, specifically the episodic Natural Actor-Critic (eNAC), known for its computational complexity of $O(\xi(d, n_t))$ per step. Here, n_t denotes the number of trajectories acquired for the current environment of the visited device. The update of $\boldsymbol{L}$ involves matrix and vector multiplications, resulting in a complexity of $O(d^3 h^2)$ for each update. The update of $\boldsymbol{s}_y$ necessitates solving a Lasso instance,

Algorithm 4 Overview of the proposed algorithm

Require: $T \leftarrow 0$, $\boldsymbol{A} \leftarrow \mathrm{zeros}_{d \times h, d \times h}$, $\boldsymbol{b} \leftarrow \mathrm{zeros}_{d \times h, 1}$
Require: $\boldsymbol{L} \leftarrow \mathrm{zeros}_{d,h}$, $\boldsymbol{D} \leftarrow \mathrm{zeros}_{d_z,h}$
Require: $\boldsymbol{\alpha}_i$, $\boldsymbol{s}_i$ for all devices
Require: $\varsigma_0 \leftarrow \mathrm{zeros}_{N+2}$, $\pi_{\boldsymbol{\theta}a}$
Stage I: IoT Devices Optimization
while some devices are not visited yet **do**
  UAV selects a random destination i
  Update $\boldsymbol{L}$, $\boldsymbol{D}$ and $\boldsymbol{s}_i$ for device i using Algorithms 1 and 2
  Update the set of visited and unvisited devices
end while

Stage II: UAV Optimization
while $t \leq T$ **do**
  UAV selects a device and velocity based on ς_m and $\pi_{\boldsymbol{\theta}_a}$
  UAV flies to destination and collects $\boldsymbol{\tau}$ and identify $\phi(\tilde{\boldsymbol{z}})$

  if $\chi_i = 1$ **then**
    Obtain $\boldsymbol{s}_{i,t}^*$ using Algorithm 3
    Obtain $\boldsymbol{\theta}_{i,t}^* = \boldsymbol{L}\boldsymbol{s}_{i,t}^*$
  else
    Compute $\boldsymbol{\theta}_{i,t}^*$ using regular PG
  end if
  Transmit the updated policy $\boldsymbol{\theta}_{i,t}^*$ to device i
  UAV computes reward $R^u(\varsigma_m, \boldsymbol{\zeta} m)$
  UAV updates $\pi_{\boldsymbol{\theta}_a}$ and $\pi_{\boldsymbol{\theta}_c}$ using (34)
  Update: next state ς_{m+1} of the UAV and $\varsigma_m \leftarrow \varsigma_{m+1}$
  Update: $t \leftarrow t + (\|\boldsymbol{l}_{m+1} - \boldsymbol{l}_m\|)/v_m$ and $m \leftarrow m + 1$
end while

typically with a complexity of $O(d^3 + hd^2 + dh^2)$ [41]. Similarly, updating the feature basis $\boldsymbol{D}$ through coupled dictionaries has a complexity of $O(h^2(d+d_z)^3)$. The update step described above is executed iteratively in the training process until all IoT devices are visited, typically requiring 80 to 120 iterations until convergence. Regarding the learning of UAV trajectories, the complexity of the Actor-Critic (AC) algorithm is $O(\epsilon^{2.5})$ [42]. The described update step is repeated during the training of the AC network while the UAV flies around, typically over approximately 25 flights. Consequently, the overall computational complexity of the proposed algorithm is calculated as $O(h^2(d + d_z)^3) + O(\xi(d, n_t) + \epsilon^{2.5})$.

D. Computational Energy

Comparing our proposed lifelong RL solution to other benchmarks, we observe that the benchmark algorithm achieves 11.41% energy saving in terms of average CPU energy consumption. However, this comes at the expense of suboptimal interacting policies for IoT devices, resulting in additional AoI cost. Overall, our proposed algorithm achieves a better comprehensive framework to balance energy and AoI. Particularly, it achieves a 14.69% improvement in average reward over the benchmark. Additionally, we compare the proposed algorithm with the two benchmarks regarding the energy consumption of the UAV. Our algorithm achievs sav-

Table II: Propulsion energy parameters of the UAV.

Parameters	Simulation Value
P_0	23.661
P_i	88.627
v_{tip}	120 m/s
v_0	4.03
d_0	0.6
s	0.05
ρ	1.225 kg/m^3
A	0.503 m^3

ings of 47.55% and 50.44% compared to the two benchmarks, respectively.

V. Simulation Results and Analysis

A. Simulation Environment and Settings

In our simulations, we consider $N = 6$ IoT devices distributed within a square region with a side length of 1km. The horizontal and vertical sides of the region are aligned with the x and y axes, respectively. We consider the use of Mica2 chips [43] on IoT devices. As such, the corresponding parameters are adopted such that $\epsilon_{\max} \in [3\times10^6, 8\times10^6]$ cycles/slot, and $\kappa_i = 10^{-21}\text{J/cycles}^3$. Moreover, the packet size $a_{i,t}$ follows a Gaussian distribution with mean $\bar{a}_{i,t} \in [1\times10^7, 5\times10^7]$ cycles and standard deviation $\sigma_{i,t} = 5\times10^6$ cycles for each environment. In addition, we consider that the duration of each environment follows a normal distribution with unique parameters for each device ranging from 100 to 550 timeslots. Moreover, each duration of a time slot is considered to be $1\,\text{s}$. In addition, we consider 1500 episodes while having each episode corresponding to 3000 timeslots. The initial location of the UAV is at the origin of the square region, i.e., $\boldsymbol{l}_0 = (0, 0)$. We consider the range of the flying velocity of the UAV to be $10\,\text{m/s} \leq v_m \leq 40\,\text{m/s}$. The rest of the propulsion energy parameters of the rotary-wing UAV are found in Table II.

More simulation details are given as follows. The dimensions of the policy $\boldsymbol{\theta}_{i,t}$ are determined by the MDP model of each environment. In our simulation, the state space has a size of $d = 2$. Accordingly, the dimension of the knowledge base $\boldsymbol{L}$ has $d = 2$ components. The feature basis $\boldsymbol{D}$ is composed of $d_z = 5$ latent components, which is equal to the dimension of the environment parameters as defined in Definition 1. The value of h, which is the dimension of each component in the knowledge base $\boldsymbol{L}$ and feature basis $\boldsymbol{D}$, is obtained from the cross-validation experiment. In our simulation, the size of h is 7. For the AC network, we employ a three-layered neural network comprising 128 neurons with both action and value heads. The Rectified Linear Unit (ReLU) activation function is applied. Given the well-established nature of the AC network, details are suppressed here since it has been explored extensively in the literature with numerous examples available.

1) Training and Testing Procedure: On the one hand, the training procedure of the proposed algorithm includes the UAV randomly flying between the devices to update the policy of each device sequentially so that it learns the knowledge base and the feature base. This process continues until all devices are visited and all environments are experienced. The knowledge base and the feature base are then updated and refined with each interaction between the UAV and devices.

On the other hand, the devices and their associated environments are considered to be randomly chosen. The UAV, equipped with its trained knowledge base and feature base, visits these devices randomly to update their policies using the zero-shot method. Following the training of the knowledge base and feature base, the trajectory of the UAV is learned using an AC network. The policy update method for all devices remains the same, while the AC network learns and optimizes the trajectory of the UAV based on the available information and feedback from the devices. The AC network has two hidden layers. The action head applies Softmax to yield the probability of the UAV visiting the devices at its next destination. A clamp function is used to capture the velocity of the UAV within the designated velocity bounds. Additionally, the value head captures the loss in the policy. Herein, we adopt zero grad as the default optimizer.

2) Benchmarks and Baselines: We consider multiple benchmarks to compare with our proposed solution. Hence, our baseline for lifelong RL solution comprises the regular *policy gradient (PG) method*. Here, instead of using the zero-shot method to enable a warm-start policy, we consider the regular PG as the base learner for each environment. In principle, any PG methods capable of estimating Hessian matrix can be incorporated, such as REINFORCE [44] and Natural Actor Critic (NAC) [45]. In our simulation, we adopt the NAC method, which is known for providing a more efficient and stable learning process. This is due to the fact that NAC incorporates natural gradients derived from information geometry, thereby respecting the underlying structure of the parameter space. For conciseness, we refer to this method as regular PG.

Furthermore, we compare our proposed AC network used for optimizing the UAV with the following baselines:

- *Random method*: The UAV chooses its flying destination randomly and flies at constant velocity of $20\,\text{m/s}$.
- *Force method*: We determine the visit interval for each device by considering the frequency of change in the device's environments. Similar to the random method, the flying velocity is set to $20\,\text{m/s}$.
- *Value-based method*: We consider a two-layer neural network for Q-learning to acquire the flight trajectories of the UAV.

B. Lifelong RL vs. Regular PG

In this subsection, we compare our proposed lifelong learning method with the Regular PG method to experiment with the effectiveness of our proposed algorithm. To do so, we ran our proposed algorithm on single environments to test its ability to generate warm start policies. Additionally, we present a comparison in continuous environments to examine its performance in non-stationary environments.

Fig. 7 compares our proposed zero-shot lifelong learning method with the regular PG method on a set of new environments. From Fig. 7, we can see that four independent environments are presented as examples to validate our approach. Unlike regular algorithms that need to start with a random initial policy, our method can provide a better warm-start policy due to the knowledge transfer incorporated in zero-shot lifelong learning. With this improved starting policy, the convergence time is greatly improved. In particular, our proposed method converges faster than the regular PG method, achieving a 50% improvement in the best case and a 25% improvement in the worst scenario case. In addition, our approach yields a 10% improvement in average reward at the beginning of an environment compared to the random initial policy. It's important to emphasize that in Fig. 7(d), our approach attains the global optimum by harnessing accumulated knowledge, a performance significantly superior to that of the baseline algorithm, which merely reaches the local optimum.

Fig. 8 shows the capability of our proposed lifelong solution in comparison to the PG method under a series of sequential environments encountered by a single IoT device. We note that the initial policy for both of the aforementioned methods is randomly generated and identical. From Fig. 8, our proposed algorithm provides better warm-start policy such that the starting reward is 33.7% higher than that achieved by regular PG method. Furthermore, it is also evident that the mean average reward of our proposed algorithm surpasses that of the regular PG method by 8.3%. This improvement can be attributed to the superior warm-start policy, which effectively reduces the convergence time. Clearly, the average reward demonstrates ongoing improvement, even as new environments continuously emerge. This underscores the sustainability of our lifelong learning algorithm as an evolving algorithm.

C. On the Impact of Lifelong RL on IoT Devices

In this subsection, we focus on the influence of our proposed lifelong RL algorithm on the system, particularly the IoT devices. Specifically, we present the impact of different non-stationary environmental periods on the average reward of IoT devices and the visiting frequency by the UAV in Fig. 9. In Fig. 10, we provide a detailed analysis of the IoT devices' performance, including AoI, CPU energy consumption and queue length.

Fig. 9 shows an example of the visits performed by the UAV to IoT devices with different environmental periods [1]. This example includes 5 devices where the duration of each environment is constant for each device. Here, the duration of each environment increases incrementally from 40 to 440 time slots as we as we transition from device 1 to device 5, respectively. From Fig. 9, it is evident that the UAV visits the IoT devices successively at different rates. In fact, as the duration of the environment increases, the likelihood that the UAV visits the corresponding IoT device decreases. In particular, 39% of the UAV's visits are allocated to device 1 while only 5% of the its visits are designated for device 5.

[1]For simplicity, we now utilize the absolute value of the reward function from here on. Hence, the rewards in upcoming figures will be represented as positive values.

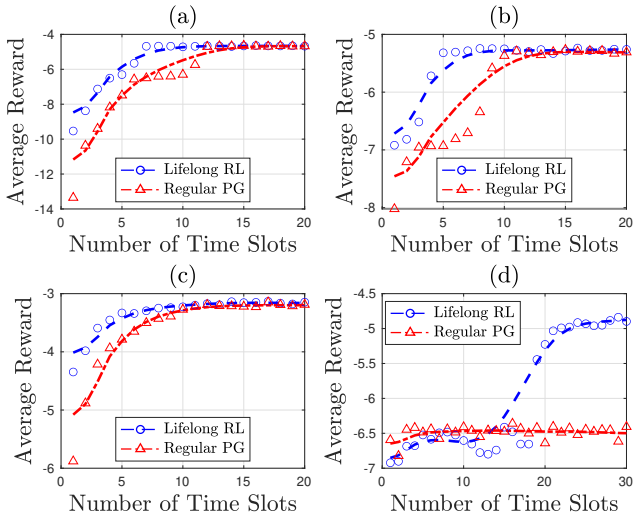

Fig. 7: Average reward of the regular PG and our proposed zero-shot lifelong learning method for 4 different examples.

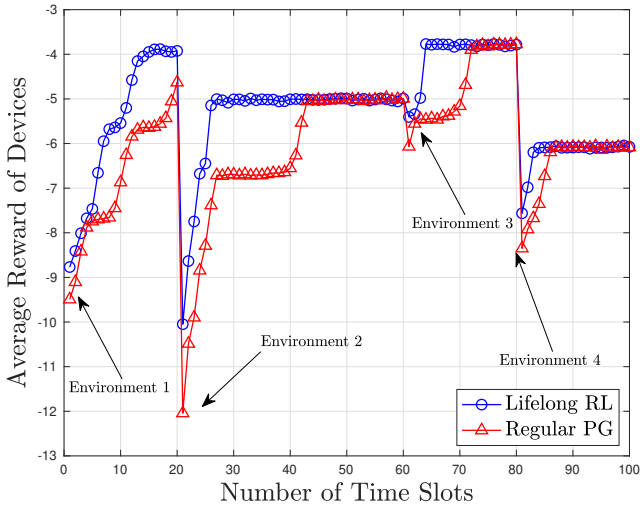

Fig. 8: Sequential learning for the PG method and zero-shot lifelong learning method over 4 environments.

This distribution aligns with the frequency of changes in the environment of each IoT device. As such, the averaged value of the reward for device 5 is 6.7212 which is remarkably better than that of device 1 having a value of 9.2810. This is due to the frequent changes in the environment that severely degrade the rewards and necessitate additional visits by the UAV to compensate for frequent policy updates. From Fig. 9(f), we can observe that the UAV responds to the environmental changes within a consistent time interval percentage across all devices. This response time accounts for 54.03% of the period for device 1, which slightly decreases to 44.44% for device 5. It is evident that the UAV exhibits accurate control over its timing for visits to the devices with respect to each environment. This is pertained by the proposed AC network, which empowers the UAV to effectively balance the disparities introduced by varying environmental periods.

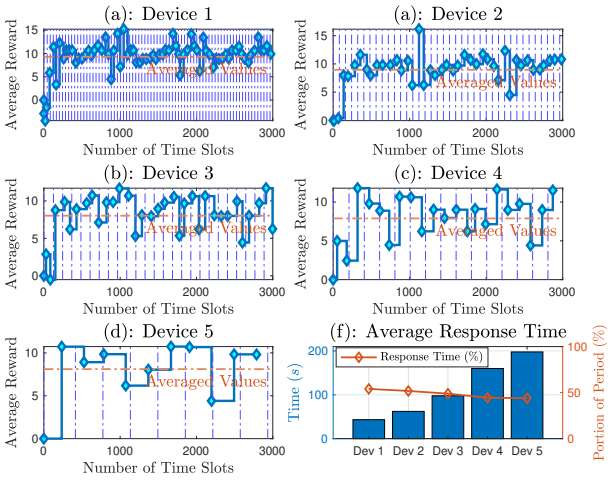

Fig. 9: History of visits performed by the UAV to 6 devices having different environmental periods.

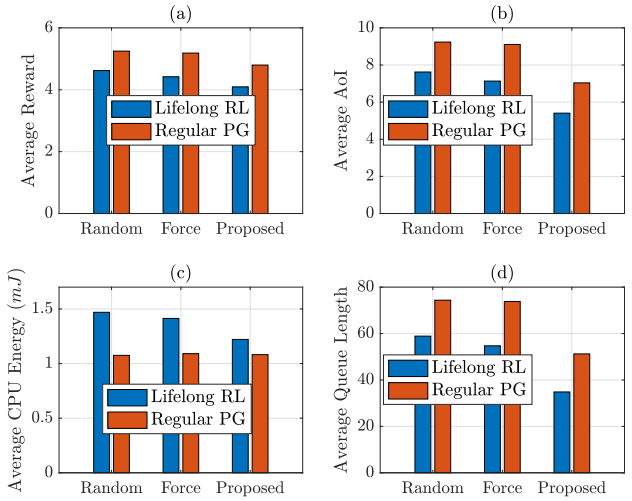

Fig. 10: Comparison for the average (a) reward, (b) AoI, (c) consumed CPU energy, and (d) queue length between the proposed lifelong RL method and the regular PG method.

Fig. 10 provides a thorough comparison between our lifelong RL method and the regular PG method for different UAV flying strategies. This comparison is carried out on multiple fronts that include the average reward, AoI, and CPU energy consumption, and queue length. On the one hand, our proposed AC method for the UAV outperforms the random and force methods on all of the aforementioned levels. On the other hand, our lifelong RL solution achieves a remarkable upper-hand in comparison to the regular PG method. In particular, from Fig. 10(a), the average reward of the lifelong RL is 14.69% lower than regular PG. This is attributed to the ability of the zero-shot method to facilitate a warm-start policy for each new environment, whereby the the duration of suboptimal policies is effectively reduced. In addition, Fig. 10(b) shows that the proposed lifelong RL solution acheives a 21.42% reduction in AoI in comaprison to the regular PG method. In addition, Fig. 10(d) shows that the lifelong RL method can achieve a reduction of 47.04% in queue length compared to regular PG. This gain arises is due to the efficient utilization of the allocated CPU resources to process the arrived data packets at each IoT device. Unlike the trends of the reward, AoI, and queue length, Fig. 10(c) showcases that the regular PG consumes less CPU energy compared to lifelong RL solution. Nevertheless, this comes at the expense of employing suboptimal interacting policies, which in turn fail to strike a balance between AoI and CPU energy consumption. This subsequently leads to increased values of rewards, AoI, and queue length. From Figs. 10(a) to 10(d), it is evident that our proposed method outperforms the PG method in terms of the total reward of devices, AoI and queue length. However, this improvement comes at the cost of increased CPU energy consumption due to the inherent trade-off between AoI and CPU energy consumption.

D. Influence of AC on UAV & IoT Devices

In order to evaluate the impact of AC, we consider the influence of the proposed algorithm on the entire UAV and IoT devices system. In Fig. 11, we evaluate the performance of the UAV and IoT devices across different UAV flying strategies. In Fig. 12 and 13, we analyze the influence of the UAV's velocities on the integrated performance of the UAV and IoT devices, respectively. In Fig. 14, we present the impact of the number of devices on the AoI and CPU energy consumption of IoT devices.

Fig. 11 showcases how the proposed AC method outperforms the random and force methods in terms on the rewards and energy efficiency for the UAV. From Fig. 11, our proposed method attains around 49% improvement in terms of system rewards with respect to the other baselines. One of the main reasons for this is the precise selection of destination and velocity that results in a significant reduction in the UAV's energy consumption that reaches 2292 kJ. This corresponds to a substantial energy savings of 48.5% as compared to the force method and random methods. From a different perspective, the rewards of the devices that are incorporated in the trade-off controlled by μ play a significant role in enhancing the overall reward of the system. Indeed, the AC method empowers the lifelong RL by equipping it with prudent flying decisions, thereby mitigating losses pertaining to suboptimal interacting policies and enhancing the rewards of the devices compared to the other two methods. These results verifies that the AC method can learn the varying distribution of environmental periods across multiple devices. Meanwhile, the force method effectively mitigates the decrease in rewards of the devices caused by suboptimal policies upon comparison to the random method. However, due to its constant velocity and lack of energy consideration, the overall reward of the force method remains inferior to our proposed approach.

Fig. 12 shows the system performance in terms of system and device rewards across different UAV flying methods for different velocities of the UAV. From Fig. 12, the energy consumption of the UAV decreases at first, after which it increases

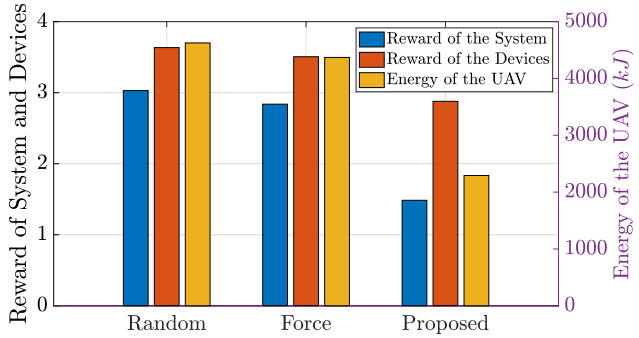

Fig. 11: The rewards and energy consumption of the random, force, and proposed AC methods for the UAV flying strategies.

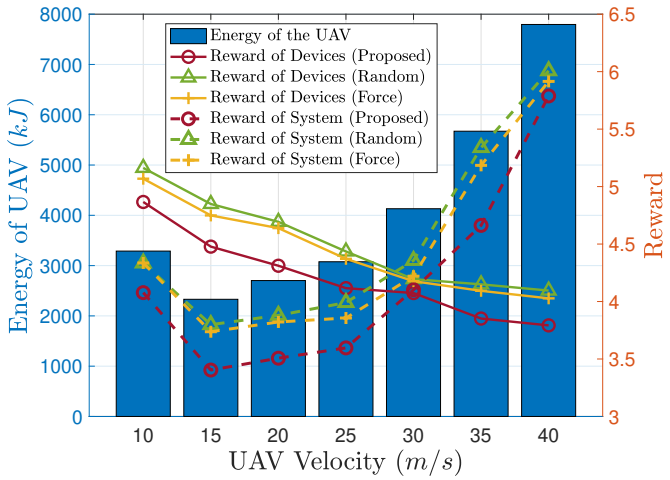

Fig. 12: The variation of the energy and rewards v.s. UAV velocity. (Left): The energy consumption of the UAV v.s. UAV velocity. (Right): The rewards of the system and devices under the proposed lifelong RL methods in comparison to the Random and Force methods v.s. UAV velocity.

with the incermental increase of the velocity from $10\,\mathrm{m/s}$ to $40\,\mathrm{m/s}$. This is consistent with the energy consumption model of the rotary wing UAV [32]. In addition, the reward of the devices decreases 22.06% as the velocity increases from $10\,\mathrm{m/s}$ to $40\,\mathrm{m/s}$. Here, maintaining a higher UAV velocity enables more frequent device visits, which can facilitate the development of improved interacting policies. Meanwhile, the force and random methods have a similar performance. In fact, the reward of the devices in the random and force method are on average 5.78% and 4.61% higher than our proposed AC method, respectively. Clearly, it is evident from Fig. 12 that the system's reward is proportional to the energy consumption of the UAV. This is reflected in a remarkable 70.05% difference between the highest and lowest rewards. Clearly, the UAV's flying energy significantly impacts the overall reward of the system. This underscores the significance of our proposed method in learning and optimizing the flying strategy of the UAV.

Fig. 13 illustrates the relationship between the AoI and CPU energy consumption of devices for different UAV velocities and for two methods: a) the proposed AC method and the b) the value-based method. From Fig. 13, the AoI in the proposed method decreases by 26.30% as the velocity increases from $10\,\mathrm{m/s}$ to $40\,\mathrm{m/s}$. This is due to the fact that allowing the UAV to conduct frequent visits enables the devices to improve their interacting policies through lifelong RL. Accordingly, this results in an increased energy demand to process incoming packets at the devices. As such, the energy consumption increases from 0.99 mJ to 2.29 mJ to account for the additional CPU processing of the devices. Subsequently, this decreases the AoI and shortens the queue lengths. This decrease is a direct implication of utilizing additional CPU energy to reduce queue length, and simultaneously, enhancing the data freshness.

Fig. 14 shows the performance of the devices as the number of the IoT devices varies from 5 to 30. The performance is comprising of their averaged AoI and CPU energy consumption, on one hand, and the reward and queue length, on the other hand. From In fact, as the number of devices increases, the UAV can perform a lower number of visits for each device. Hence, suboptimal interacting policies that degrade the performance of each device arise accordingly. Thus, from Fig. 14(a), we can observe that the AoI increases from 4.81 to 9.81, as the number of devices increases. This observation underscores the importance of constraining the UAV's flying range to achieve optimal efficiency. In contrast, the CPU energy consumption experiences a significant decrease that reaches up to 55% as the number of devices increases. Here, the energy consumption reaches a plateau when the number of devices exceeds 10. Evidently, poor interacting policies can bring drastic effects to the energy efficiency of the CPU. From Fig. 14(b), the reward of the devices increases by 59.29% as the number of devices increases. Furthermore, the average queue length for all the devices also experiences an increase due to the delayed processing of incoming data packets. Hence, as the number of devices increases, we can see that the having many devices will inevitably lead to an increase in the rewards. However, this comes at the expense of delayed UAV response and an elevated queue length.

E. Lifelong Curriculum Learning

In this subsection, we consider the influence of lifelong curriculum learning. Unlike random environments, we selectively arrange the difficulties of environments and their corresponding orders. In Fig. 15,we evaluate the performance of lifelong RL models trained by the basic and complex environments separately. In Fig. 16, we evaluate the performance of the AC model with shuffled environment orders.

To evaluate the influence of different training environments on the performance of lifelong RL method, we consider two types of environments: basic environments and complex environments. These environments are characterized based on their convergence performance, including the convergence rate and the average reward achieved at convergence. With these different environments as the training environments, we consider three lifelong RL models. These models encompass those trained solely on basic environments (Easy Model), those trained solely on complex environments (Difficult Model), and

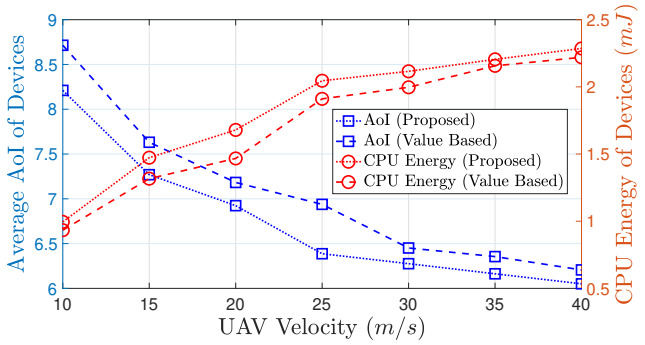

Fig. 13: The AoI and CPU energy consumption as a function of the velocity of the UAV for the proposed and value-based methods.

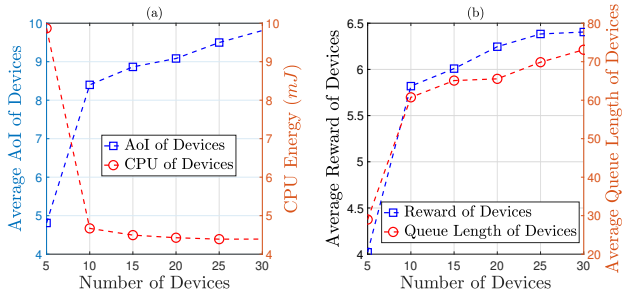

Fig. 14: The (a) AoI and CPU energy consumption, and (b) reward of IoT devices and queue length as a function of the number of devices.

those trained on a mixed training set comprising both basic and complex environments (Mix Model). Fig. 15 depicts the average reward of the three models tested on the basic and complex environments. As shown in Fig. 15(a), the average rewards of the three considered models and the regular PG model are evaluated on 40 basic environments. The performance of all the three models under consideration consistently exceeds that of the regular PG model, demonstrating an improvement of approximately 25.31% at the beginning and 5.99% upon convergence. The model trained on complex environments consistently outperforms both Easy model and Mix model throughout the learning process, with initial improvements of 24.98% and 24.48% over regular PG, respectively. This is attributed to the inherent ability of Difficult model to capture and leverage more profound knowledge, while the other two models fail to exploit the knowledge effectively. The superiority of the Easy model over the Mix model is attributed to the consistency in the training environments, facilitating a learning process with gradients converging towards the target parameter areas. In addition, it is shown in Fig. 15(a) that our proposed method converges to a better value than the regular PG. This noteworthy enhancement is attributed to the incorporation of our knowledge base, which facilitates effective exploitation beyond local optima. Fig. 15(b) shows the performance of the models specifically on complex environments. It is evident that the Difficult model significantly outperforms the Easy and Mix models, showcasing improvements of 3.36% and 4.87%, respectively. This aligns with the previously discussed observation, wherein the Difficult model's ability to capture more profound knowledge contributes to its superior performance. In

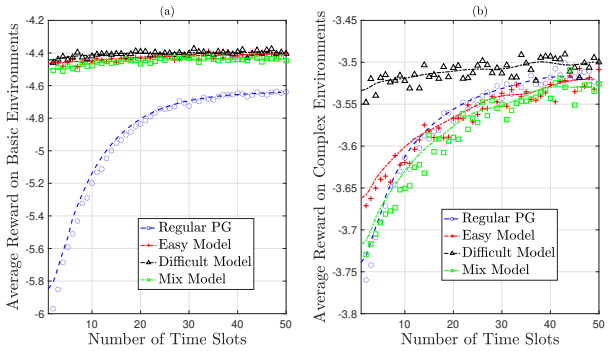

Fig. 15: The average reward of different lifelong RL models on (a) basic environments and (b) complex environments. The lines represent the smoothed average of the scattered dots.

addition, the Easy model is better than Mix model in terms of warm start policy. This comes from the fact that mixed training environments underperform the knowledge base. The reason is that it is difficult for a model with divergent training examples to capture the specific knowledge and learn a unified representation, leading to possible misderstanding of some particular type of environments. Furthermore, the disparity between the two subfigures indicates the challenges encountered by the models in capturing the features of complex environments compared to their performance in basic environments.

In order to evaluate the influence of the environmental orders on the UAV and IoT devices system, we consider four distinct shuffling scenarios:

- Scenario 1: All the environments used to train the AC model are basic environments. For each device, a collection of basic environments are experienced sequentially over time. One practical example is smart agriculture, where the environmental dynamics and parameters are relatively simple and straightforward.
- Scenario 2: All the considered environments are complex environments. For example, these complex environments represent intricate challenges encountered in intelligent robotics. Intelligent robots must process complex environmental parameters to make correct decisions regarding movement or operations. In this case, each device is exposed to a set of complex environments with random orders.
- Scenario 3: Unlike Scenario 1 and Scenario 2, a mixture of basic and complex environments is considered. Specifically, only basic environments are experienced during the first half of the time period $[0, T)$, while complex environments are experienced during the second half of the time interval. Take a smart vehicle traversing different surrounding environments as an example [46]. Simple environments, such as rural areas, are initially encountered, followed by more complex environments, such as urban cityscapes.
- Scenario 4: As a flip of Scenario 3, Scenario 4 considers the case where complex environments are followed by simple and basic environments. Similar to Scenario 3, the time interval $[0, T)$ is divided into two halves: first complex environments, followed by basic environments.

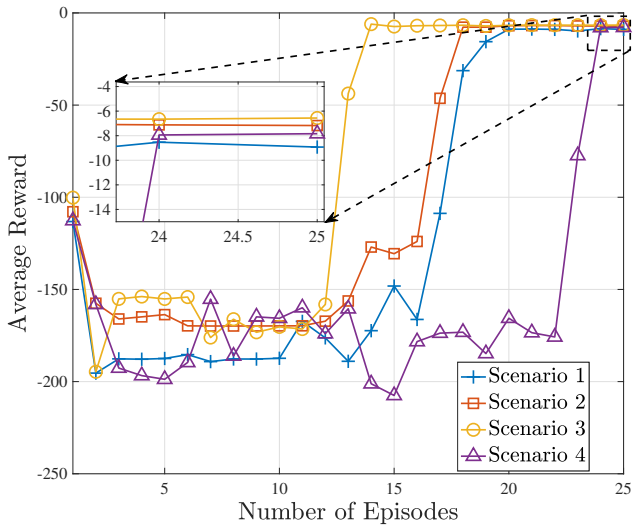

Fig. 16: Comparison of the average reward for four environments shuffling orders: Scenario 1, Scenario 2, Scenario 3 and Scenario 4.

As depicted in Fig. 16, all shuffling scenarios exhibit a decrease in average reward before an ascent, ultimately converging. Scenario 3 demonstrates the fastest convergence speed, achieving convergence within up to 14 episodes and attaining the highest convergence value among the four shuffling scenarios. The training strategy of tackling basic environments first, followed by advaced ones, as outlined in [47], can enhance model performance and expedite the learning rate. This is due to the fact that encountering basic environments initially guides the learning agent towards a more favorable parameter space, effectively mitigating the impact of noise introduced by diverse training samples. Subsequently, engaging in complex environments aids Scenario 1 in moving towards target parameter solutions with increased confidence. In Fig. 16, shuffling scenarios with homogeneous environments, such as Scenario 1 and Scenario 2, demonstrate superior convergence speed compared to Scenario 4. For reasons analogous to those in Fig. 15, homogeneous environments can guide the models with consistent gradients, while Scenario 4 introduces noises to the learning process, resulting in delayed learning. As also shown in Fig. 16, Scenario 3 attains the best convergence value, exhibiting the highest improvement of up to 24.42% compared to Scenario 1. This aligns with the expectation of guidance and denoising capabilities provided by the basic-first-then-complex order. In contrast to the impact of shuffling orders on convergence speed, Scenario 2 and Scenario 4 exhibit superior convergence values, compared to Scenario 1. In other words, the inclusion of complex environments in the training samples enhances the model's ability to exploit additional knowledge, thereby providing better insights into optimal parameters.

VI. Conclusion

In this paper, we have proposed a novel UAV-aided lifelong RL solution that leverages a UAV to adapt the policies of IoT devices in non-stationary environments. Our proposed method aims to continuously optimize the data freshness and energy efficiency of IoT devices, while efficiently utilizing the energy resources of the UAV. Hence, we have designed a lifelong RL solution that leverages a shared knowledge base and feature base to acquire the optimal policies for the dynamic non-stationary environments. In addition, we have proposed a zero-shot method to determine the warm-start policies for unseen environments. To efficiently utilize the energy resources of the UAV as it learns from the environments, its corresponding flying trajectory and velocities are optimized by adopting an AC network solution mechanism. Our simulations have validated that the proposed lifelong approach yields significant performance gains in terms of minimized AoI and optimal energy efficiency for both IoT devices and the UAV, respectively. A potential avenue for future exploration lies in investigating non-stationarity across diverse domains. In this case, the implementation of fully distributed knowledge bases has the potential to facilitate a more flexible learning structure across extensive networks of mobile devices.

References

[1] Z. Gong, Q. Cui, C. Chaccour, B. Zhou, M. Chen, and W. Saad, "Lifelong Learning for Minimizing Age of Information in Internet of Things Networks," *IEEE International Conference on Communications (ICC)*, pp. 1–6, June, 2021.

[2] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, P. Popovski, and M. Debbah, "Seven Defining Features of terahertz (THz) Wireless Systems: A Fellowship of Communication and Sensing," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 967–993, 2022.

[3] O. Hashash, C. Chaccour, and W. Saad, "Edge Continual Learning for Dynamic Digital Twins Over Wireless Networks," *arXiv preprint arXiv:2204.04795*, 2022.

[4] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time Status: How Often Should One Update?" *Proceedings IEEE INFOCOM*, pp. 2731–2735, 2012.

[6] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the Role of Age of Information in the Internet of Things," *IEEE Communications Magazine*, vol. 57, no. 12, pp. 72–77, 2019.

[7] X. Pang, N. Zhao, J. Tang, C. Wu, D. Niyato, and K.-K. Wong, "IRS-assisted Secure UAV Transmission via Joint Trajectory and Beamforming Design," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1140–1152, 2021.

[8] N. Zhao, X. Pang, Z. Li, Y. Chen, F. Li, Z. Ding, and M.-S. Alouini, "Joint Trajectory and Precoding Optimization for UAV-assisted NOMA Networks," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3723–3735, 2019.

[9] X. Pang, M. Sheng, N. Zhao, J. Tang, D. Niyato, and K.-K. Wong, "When UAV Meets IRS: Expanding Air-ground Networks via Passive Reflection," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 164–170, 2021.

[10] X. Li, Q. Cui, D. Feng, Z. Gong, and X. Tao, "Deep Reinforcement Learning-Based Solution for Minimizing the Alterable Urgency of Information in UAV-Enabled IIoT System," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 437–442.

[11] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal Trajectory Planning and Data Collection in UAV-assisted Wireless Powered IoT Networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1211–1223, 2020.

[12] A. A. Laghari, K. Wu, R. A. Laghari, M. Ali, and A. A. Khan, "A Review and State of art of Internet of Things (IoT)," *Archives of Computational Methods in Engineering*, pp. 1–19, 2021.

[13] S. Thrun, "Lifelong Learning Algorithms," in *Learning to Learn*. Springer, 1998, pp. 181–209.

[14] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in Non-stationary Environments: A Survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[15] S. Padakandla, "A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

[16] C. O'Reilly, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Anomaly Detection in Wireless Sensor Networks in a Non-stationary Environment," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1413–1432, 2014.

[17] S. Padakandla, K. Prabuchandran, and S. Bhatnagar, "Reinforcement Learning Algorithm for Non-stationary Environments," *Applied Intelligence*, vol. 50, no. 11, pp. 3590–3606, 2020.

[18] B. Zhou and W. Saad, "Joint Status Sampling and Updating for Minimizing Age of Information in the Internet of Things," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7468–7482, 2019.

[19] Q. Wang, H. Chen, Y. Gu, Y. Li, and B. Vucetic, "Minimizing the Age of Information of Cognitive Radio-based IoT Systems Under a Collision Constraint," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 8054–8067, 2020.

[20] M. Hatami, M. Jahandideh, M. Leinonen, and M. Codreanu, "Age-aware Status Update Control for Energy Harvesting IoT Sensors via Reinforcement Learning," *IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–6, Aug. 2020.

[21] A. Ferdowsi, M. A. Abd-Elmagid, W. Saad, and H. S. Dhillon, "Neural Combinatorial Deep Reinforcement Learning for Age-optimal Joint Trajectory and Scheduling Design in UAV-assisted Networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1250–1265, 2021.

[22] Q. Dang, Q. Cui, Z. Gong, X. Zhang, X. Huang, and X. Tao, "AoI Oriented UAV Trajectory Planning in Wireless Powered IoT Networks," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 884–889.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[25] A. Xie, J. Harrison, and C. Finn, "Deep Reinforcement Learning Amidst Lifelong Non-stationarity," *arXiv preprint arXiv:2006.10701*, 2020.

[26] S. Abdallah and M. Kaisers, "Addressing Environment Non-stationarity by Repeating Q-learning Updates," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1582–1612, 2016.

[27] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic Meta-learning for Fast Adaptation of Deep Networks," *International Conference on Machine Learning*, pp. 1126–1135, Sydney, Australia, Aug. 2017.

[28] A. Kumar and H. Daume III, "Learning Task Grouping and Overlap in Multi-task Learning," *arXiv preprint arXiv:1206.6417*, 2012.

[29] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image Super-resolution via Sparse Representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[30] Z. Chen and B. Liu, "Lifelong Machine Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.

[31] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS Digital Design," *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992.

[32] Y. Zeng, J. Xu, and R. Zhang, "Energy Minimization for Eireless Communication with Rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[33] H.-Y. Zhou, D.-Y. Luo, Y. Gao, and D.-C. Zuo, "Modeling of Node Rnergy Consumption for Wireless Sensor Networks," *Wireless Sensor Network*, vol. 3, no. 1, p. 18, 2011.

[34] J. Wu, S. Rangan, and H. Zhang, *Green Communications: Theoretical Fundamentals, Algorithms, and Applications*. CRC press, 2016.

[35] Q. Cui, Z. Gong, W. Ni, Y. Hou, X. Chen, X. Tao, and P. Zhang, "Stochastic Online Learning for Mobile Edge Computing: Learning from Changes," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 63–69, 2019.

[36] S. Thrun and T. M. Mitchell, "Lifelong Robot Learning," *Robotics and Autonomous Systems*, vol. 15, no. 1-2, pp. 25–46, 1995.

[37] D. L. Silver, Q. Yang, and L. Li, "Lifelong Machine Learning Systems: Beyond Learning Algorithms," *AAAI Spring Symposium Series*, pp. 49–55, 2013.

[38] Z. Yu, F. Wu, Y. Yang, Q. Tian, J. Luo, and Y. Zhuang, "Discriminative Coupled Dictionary Hashing for Fast Cross-media Retrieval," *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 395–404, Queensland, Australia, July 2014.

[39] R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng, "Zero-shot Learning Through Cross-modal Transfer," *arXiv preprint arXiv:1301.3666*, 2013.

[40] B. Romera-Paredes and P. Torr, "An Embarrassingly Simple Approach to Zero-shot Learning," *International Conference on Machine Learning*, pp. 2152–2161, Lille, France, July 2015.

[41] J. Mairal and B. Yu, "Complexity Analysis of the Lasso Regularization Path," *arXiv preprint arXiv:1205.0079*, 2012.

[42] H. Kumar, A. Koppel, and A. Ribeiro, "On the Sample Complexity of Actor-critic Method for Reinforcement Learning with Function Approximation," *Machine Learning*, pp. 1–35, 2023.

[43] J. L. Hill and D. E. Culler, "Mica: A Wireless Platform for Deeply Embedded Networks," *IEEE micro*, vol. 22, no. 6, pp. 12–24, 2002.

[44] R. J. Williams, "Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[45] J. Peters and S. Schaal, "Natural Actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[46] Q. Cui, X. Hu, W. Ni, X. Tao, P. Zhang, T. Chen, K.-C. Chen, and M. Haenggi, "Vehicular Mobility Patterns and Their Applications to Internet-of-Vehicles: A Comprehensive Survey," *Science China Information Sciences*, vol. 65, no. 11, p. 211301, 2022.

[47] X. Wang, Y. Chen, and W. Zhu, "A Survey on Curriculum Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4555–4576, 2021.

Zhenzhen Gong (Graduate Student Member, IEEE) received the B.S. degree in communication engineering from Yanshan University, China, in 2016. She is currently pursuing the Ph.D. degree in information and communications engineering with the Beijing University of Posts and Telecommunications. Her current research interests include intelligent edge networks, age of information, and the application of artificial intelligence within the Internet of Things.

Omar Hashash (Graduate Student Member, IEEE) received the B.E. degree in communications and electronics engineering from Beirut Arab University, Beirut, Lebanon, in 2019 and the M.E. degree in electrical and computer engineering from the American University of Beirut, in 2021. He is currently working toward the Ph.D. degree with Electrical and Computer Engineering Department, Virginia Tech, Blacksburg, VA, USA. He has authored or coauthored one of the first works that explore the synergy between wireless, computing, and AI techniques that can come together to support massive digital twinning of physical systems in the metaverse. His research interests include 6G wireless networks, digital twins, metaverse, edge intelligence, and generalizable AI.

Yingze Wang received the B.S. degree in communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2015 and the Ph.D degree in information and communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2023. His current research interests include the proactive communication theory, intelligent wireless networks, and the applications of reinforcement learning within resource management.

Qimei Cui (Senior Member, IEEE) received the B.E. and M.S. degrees in electronic engineering from Hunan University, Changsha, China, in 2000 and 2003, respectively, and the Ph.D. degree in information and communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2006. She has been a Full Professor with the School of Information and Communication Engineering, BUPT, since 2014. She was a Visiting Professor with the Department of Electronic Engineering, University of Notre Dame, Notre Dame, IN, USA, in 2016. Her research interests include B5G/6G wireless communications, mobile computing, and the IoT.

Prof. Cui serves as a Technical Program Committee Member of several international conferences, such as the IEEE ICC, the IEEE WCNC, the IEEE PIMRC, the IEEE ICCC, the WCSP 2013, and the IEEE ISCIT 2012. She won the Best Paper Award at IEEE ISCIT 2012, IEEE WCNC 2014, and WCSP 2019, the Honorable Mention Demo Award at ACM MobiCom 2009, and the Young Scientist Award at URSI GASS 2014. She serves as the Technical Program Chair for APCC 2018, the Track Chair for IEEE/CIC ICCC 2018, and the Workshop Chair for WPMC 2016. She serves as an Editor for Science China Information Sciences and a Guest Editor for the EURASIP Journal on Wireless Communications and Networking, International Journal of Distributed Sensor Networks, and Journal of Computer Networks and Communication.

Wei Ni (Fellow, IEEE) received the B.E. and Ph.D. degrees in Electronic Engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively. He is a Principal Research Scientist at CSIRO, and a Conjoint Professor at the University of New South Wales, Sydney, Australia. He is also an Adjunct Professor at the University of Technology Sydney, and an Honorary Professor at Macquarie University. He serves as a Technical Expert at Standards Australia in support of the ISO standardization of AI and Big Data. He was a Postdoctoral Research Fellow at Shanghai Jiaotong University from 2005 to 2008; Deputy Project Manager at Bell Labs, Alcatel/Alcatel-Lucent from 2005 to 2008; and Senior Researcher at Devices R&D, Nokia from 2008 to 2009. He has co-authored one book, ten book chapters, more than 300 journal papers, more than 100 conference papers, 26 patents, ten standard proposals accepted by IEEE, and three technical contributions accepted by ISO. His research interests include 6G security and privacy, machine learning, stochastic optimization, and their applications to system efficiency and integrity.

Dr. Ni has been an Editor for IEEE Transactions on Wireless Communications since 2018, an Editor for IEEE Transactions on Vehicular Technology since 2022, and an Editor for IEEE Transactions on Information Forensics and Security and IEEE Communications Surveys and Tutorials since 2024. He served first as the Secretary, then the Vice-Chair and Chair of the IEEE VTS NSW Chapter from 2015 to 2022, Track Chair for VTC-Spring 2017, Track Co-chair for IEEE VTC-Spring 2016, Publication Chair for BodyNet 2015, and Student Travel Grant Chair for WPMC 2014.

Walid Saad (S'07, M'10, SM'15, F'19) received his Ph.D degree from the University of Oslo, Norway in 2010. He is currently a Professor at the Department of Electrical and Computer Engineering at Virginia Tech, where he leads the Network sciEnce, Wireless, and Security (NEWS) laboratory. His research interests include wireless networks (5G/6G/beyond), machine learning, game theory, quantum communications/learning, security, UAVs, semantic communications, cyber-physical systems, and network science. Dr. Saad is a Fellow of the IEEE. He is also the recipient of the NSF CAREER award in 2013, the AFOSR summer faculty fellowship in 2014, and the Young Investigator Award from the Office of Naval Research (ONR) in 2015. He was the (co-)author of twelve conference best paper awards at IEEE WiOpt in 2009, ICIMP in 2010, IEEE WCNC in 2012, IEEE PIMRC in 2015, IEEE SmartGridComm in 2015, EuCNC in 2017, IEEE GLOBECOM (2018 and 2020), IFIP NTMS in 2019, IEEE ICC (2020 and 2022), and IEEE QCE in 2023. He is the recipient of the 2015 and 2022 Fred W. Ellersick Prize from the IEEE Communications Society, of the IEEE Communications Society Marconi Prize Award in 2023, and of the IEEE Communications Society Award for Advances in Communication in 2023. He was also a co-author of the papers that received the IEEE Communications Society Young Author Best Paper award in 2019, 2021, and 2023. Other recognitions include the 2017 IEEE ComSoc Best Young Professional in Academia award, the 2018 IEEE ComSoc Radio Communications Committee Early Achievement Award, and the 2019 IEEE ComSoc Communication Theory Technical Committee Early Achievement Award. From 2015-2017, Dr. Saad was named the Stephen O. Lane Junior Faculty Fellow at Virginia Tech and, in 2017, he was named College of Engineering Faculty Fellow. He received the Dean's award for Research Excellence from Virginia Tech in 2019. He was also an IEEE Distinguished Lecturer in 2019-2020. He has been annually listed in the Clarivate Web of Science Highly Cited Researcher List since 2019. He currently serves as an Area Editor for the IEEE Transactions on Communications. He is the Editor-in-Chief for the IEEE Transactions on Machine Learning in Communications and Networking.

Kei Sakaguchi (Senior Member, IEEE) received the M.E. degree in information processing and the Ph.D. degree in electrical and electronics engineering from theTokyo InstituteTechnology,Tokyo, Japan, in 1998 and 2006, respectively. He is currently theDean with the Tokyo Institute of Technology, a Professor with the Tokyo Tech Academy for Super Smart Society, and also a Professor with the School of Engineering. He is currently the Outside Director with oRo Company, Ltd., Osaka, Japan. His current research interests include 5G cellular networks, millimeter-wave communications, wireless energy transmission, V2X for automated driving, and super smart society. Dr. Sakaguch was the recipient of the Outstanding Paper Awards from SDR Forum and IEICE, in 2004 and 2005, respectively, three Best Paper Awards from IEICE communication society in 2012, 2013, and 2015, respectively, and the Tutorial Paper Award from IEICE communication society in 2006. He is a Fellow of IEICE.