IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 12 March 2024; revised 19 July 2024 and 20 September 2024; accepted 14 November 2024.
Date of publication 18 November 2024; date of current version 5 December 2024.

The associate editor coordinating the review of this article and approving it for publication was Z. Yang.

Digital Object Identifier 10.1109/TMLCN.2024.3501217

A2PC: Augmented Advantage Pointer-Critic Model for Low Latency on Mobile IoT With Edge Computing

RODRIGO CARVALHO[1], FAROQ AL-TAM[2], AND NOÉLIA CORREIA[3]

[1]Center for Electronic, Optoelectronic and Telecommunications (CEOT), University of Algarve, 8005-139 Faro, Portugal
[2]ELM-Research, Riyadh 12382, Saudi Arabia
[3]Faculty of Science and Technology, Center for Electronic, Optoelectronic and Telecommunications (CEOT), University of Algarve, 8005-139 Faro, Portugal

CORRESPONDING AUTHOR: R. CARVALHO (rzcarvalho@ualg.pt)

This work was supported in part by the Foundation for Science and Technology (FCT) from Portugal with the Center for Electronic, Optoelectronic and Telecommunications (CEOT's) through CEOT BASE under Grant UIDB/00631/2020; and in part by the CEOT PROGRAMÁTICO Project UIDP/00631/2020.

ABSTRACT As a growing trend, edge computing infrastructures are starting to be integrated with Internet of Things (IoT) systems to facilitate time-critical applications. These systems often require the processing of data with limited usefulness in time, so the edge becomes vital in the development of such reactive IoT applications with real-time requirements. Although different architectural designs will always have advantages and disadvantages, mobile gateways appear to be particularly relevant in enabling this integration with the edge, particularly in the context of wide area networks with occasional data generation. In these scenarios, mobility planning is necessary, as aspects of the technology need to be aligned with the temporal needs of an application. The nature of this planning problem makes cutting-edge deep reinforcement learning (DRL) techniques useful in solving pertinent issues, such as having to deal with multiple dimensions in the action space while aiming for optimum levels of system performance. This article presents a novel scalable DRL model that incorporates a pointer-network (Ptr-Net) and an actor–critic algorithm to handle complex action spaces. The model synchronously determines the gateway location and visit time. Ultimately, the gateways are able to attain high-quality trajectory planning with reduced latency.

INDEX TERMS Action branching, Internet of Things, IoT, long-range wide-area network, LoRaWAN, mobility, pointer networks, reinforcement learning.

I. INTRODUCTION

RECENT advances in Internet of things (IoT) networks have led to many changes in a variety of industrial domains. Among these is the agricultural industry, where precision technology and the timely adjustment of processes have decisive impacts on business success [1]. Managing the massive deployment of sensing devices in these types of networks is a bottleneck; hence, the development of artificial intelligence (AI) solutions is a high priority.

Mobile gateways, including terrestrial and unmanned aerial vehicles (UAV), have been efficiently used in numerous network environments that connect hard-to-reach areas and regions of occasional data generation [2]. The constrained nature of remote-area systems and the demand for timely data processing entail solutions that combine mobile gateways with edge-storage technologies [3]. Edge computing enables reduced latency and lower workloads via cloud edge-computing tradeoffs. However, for these systems to work successfully, the gateway collection and storage systems must be properly synchronized, and networking mobility must be jointly scheduled and optimized to match the user needs of data freshness.

Long range wide area networks (LoRaWAN) technologies are among the most successful stacks that have been applied in the automation of IoT solutions, with increasing popularity in agricultural solutions [4]. However, despite the urgent need, the use of mobile LoRaWAN gateways with edge support has not been fully explored.

This article proposes a framework leveraging a state-of-the-art machine-learning model and a mathematical optimization model, that together produce a functional pipeline to effectively address the multidimensional LoRaWAN mobility problem with edge systems. The proposed deep reinforcement learning (DRL) agent maximizes its reward based on an efficient trajectory while respecting data expiration times and location visit waiting times.

A. PROBLEM DEFINITION

Mobile LoRaWAN gateways have been considered as cost-effective solutions to the otherwise wasteful approach of covering vast areas with static gateways [5]. Static networks require fixed gateway positioning, signal overlapping inspections, and considerable battery consumption [1]. However, if the gateways are allowed to move, these issues can be relatively mitigated, particularly when strategic edge infrastructural maneuvering is applied (Figure 1). Therefore, proper gateway mobility planning is essential.

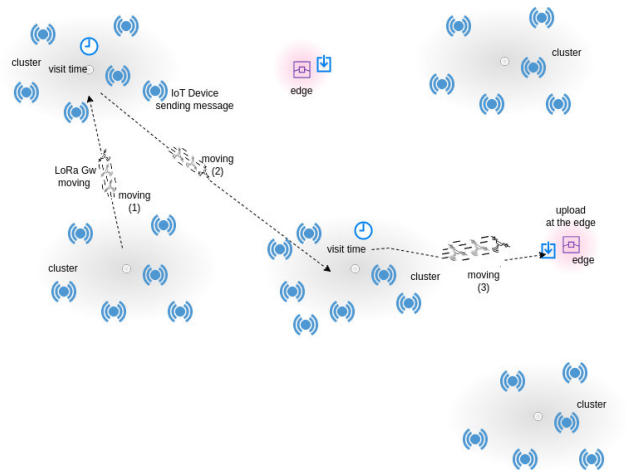

FIGURE 1. Desired edge-supported architecture.

This article addresses the LoRaWAN gateway mobility planning problem using an edge system cooperation scheme based on visit time planning (GMEVTP) and its optimization. As shown in Figure 1, the physical network topology is represented by clusters, where each cluster contains either LoRaWAN devices that send data or edge systems that collect, store, and operate on them. The gateway initially moves to the center of a cluster and remains there for a period while collecting or uploading data. The physical landscape could be visualized as a grid map [6], however, in this work, the clustering approach [7] is used as it enables better-mapped cluster simulations. luster communication settings are determined using a custom reinforcement learning (RL) adaptive data-rate approach [8].

To address gateway mobility and visit times simultaneously, we propose a novel augmented advantage pointer critic (A2PC) model that makes use of a pointer neural network with two heads. One estimates the location, while the other estimates the visit time. To train this new architecture, an advantage actor–critic (A2C) loss function is provided. Neural network branching has been successfully used to address issues in several areas [9]. However, to our knowledge, this is the first instance of its use in a scenario that includes LoRaWAN and mobility planning. Furthermore, the overall framework, architecting different optimization models, provides a unique and innovative approach to the addressed network design problem. In general, the approach in this proposal is in line with the current adopted techniques for network management, such as self-driving networks [10]. Although alternative mobility planning solutions exist, such as IoT-Mobile [11] and gateway mobility with edge system (GMwES) [12], solving the end-to-end mobility and visit time estimation problem using a DRL solution has not been explored yet. This type of solution is needed to address the inherited NP-hard scheduling problem [13]. Notably, most contemporary statistical regression and mathematical models cannot fully capture the necessary environmental features or become unfeasible. Hence, machine-learning approaches are leveraged.

B. CONTRIBUTIONS

The below enumeration provides a concise overview of the accomplishments derived from this work:

- A mathematical optimization model (MIPOPT) of the GMEVTP optimization problem is provided.
- A novel end-to-end-trained DRL architecture capable of estimating gateway locations and visit times simultaneously is produced.
- A data generation simulation scheme that utilizes the NS3 discrete event network simulator is used to simulate communication networks.
- A detailed evaluation with empirical key performance indicators (KPIs) are provided.

The remainder of this article is organized as follows. Section II discusses recent studies related to DRL techniques and mobile/edge LoRa solutions. Section III describes the MIPOPT, our proposed A2PC agent, and the entire framework for solving the given problem. The details of the experiment, performance, and analysis are presented in Section IV. Finally, Section V concludes the article and discusses future research.

II. LITERATURE REVIEW

Interest in mobile LoRaWAN continues to grow towards efficient service provisions based on mobility and workload distribution. In line with that, machine-learning approaches become more effective in solving network management problems. This trend results in studies that articulate these areas of knowledge together. Hence, some of the relevant state-of-the-art literature are listed in this section.

A. LORA, UAV, AND GATEWAY MOBILITY

LoRaWAN path planning and mobile gateway deployment may require edge computing to serve as storage systems. That

is addressed in a previous work [12], where the objective maximizes the throughput and processing time by optimizing travel hops and edge line-of-sight communications. That approach is reflected in visit and loitering times based on static schedules. Furthermore, action and state-space patterns were hardcoded into the architecture, and changes required full retraining of the machine-learning agents. Although the mobility planning problem was addressed in [12], the current study's proposal approaches the solution from a unique and improved DRL perspective. Previous studies [5], [14] discussed the use of LoRaWAN in farming applications, where a mobile gateway was used to reduce energy consumption and overall costs. The objective was to increase the productivity of livestock management systems. However, the solutions did not directly address mobility planning (i.e., scheduling was reactionary). In [15], the authors discuss efficient resource allocation for distributed reconfigurable intelligent surfaces (RISs) assisted probabilistic semantic communication (PSC), with high potential in wireless systems.

Concerning mobility, UAVs are engineered with control systems to guarantee specific mission profiles, which often include linking distant communication nodes and end users. For UAVs to function well as the key connectors of end users and devices to larger networks, they must possess the required hardware, power, and communication capabilities. Moreover, they must have the ability to sufficiently cover the designated geographical area [16], [17], [18], [19], [20], [21], [22], [23], [24], [25].

A recent study [26] combined the travel salesman problem (TSP) with particle swarm optimization to improve data gathering from multiple sensors. However, the proposal does not address latency issues or edge-system solutions. To support multihopping protocols and mobile LoRaWAN gateways, [27] presented an algorithm that enables emergency systems to leverage LoRaWAN networks whenever global positioning systems (GPS) and cellular services are insufficient. However, the study did not address latency concerns. The work described in [28] introduced an energy-efficient opportunistic model that employed ant-based routing algorithms to optimize large-scale sensor networks. However, the primary emphasis was on the collection of energy efficiency data, which did not help resolve end-to-end operations. Mobile LoRaWAN gateways were employed in [29] to assist firefighter systems during static gateway outages. A time-slotted transmission scheduling protocol was used, and the results demonstrated that LoRaWAN solutions are well-suited for time-sensitive scenarios. The authors of [30] conducted a systematic review in which they classified known problems and proposed solutions based on assumptions related to mobile sensor networks. Their conclusions and conventions provide directions for the current research.

B. LORA AND EDGE SYSTEMS INTEGRATION

Integrating edge computing with LoraWAN solutions requires resolving some obstacles. In [31], uplink solutions were used to deliver extension capabilities to improve LoRaWAN architectures via the addition of multiple hops. The design of suitable data delivery protocols is also important, and was addressed in [32]. The study proposed a lightweight long-range protocol that facilitates edge data delivery while enabling improved throughput and lower error rates. In [33], an optimization problem is set up to resolve resource allocation in a probabilistic semantic communication network, and the potential of integrating semantic communication with mobile edge computing is discussed. A comprehensive study on the exploration of edge learning strategies has been presented in [34]. The techniques focus on the development of dual-functional designs that may enhance both communication optimization and the learning models deployed at the edge nodes. The findings show that investigating edge learning might be advantageous for meeting application requirements. While the output emphasizes the achievements in the 5G ecosystem, the underlying principles may also be applied to other use cases, such as LoRaWAN. In [3], an edge-assisted IoT proof-of-concept using LoRa was presented, and useful guidelines for integrated IoT ecosystems were provided. The work in [35] addressed the use of moving gateways while considering travel times and distance minimization. Although such a proposal covered trajectory planning, it was designed to delivery data loads only once, at the end of its circuitous travel. This need is only a subset of the whole problem discussed in the present study.

C. DRL-BASED SOLUTIONS

In the LoRaWAN context, a few specific techniques are known to be beneficial to the problem at hand. First, dueling networks [36] are used to generate distinct branches from a primary neural network to surpass conventional long short-term memory (LSTM) video-gaming solutions. Similarly, [9] proposed a DRL solution for solving discrete and high-dimensional action problems, and [37] provided a branching network to overcome high-dimensional decision space issues and improved the operation of a microgrid using distributed battery energy storage systems. The innovative branching–dueling Q-network (BDQ) agent is scalable in various environments and offers a strong candidate solution for network management and planning applications. Similarly, the method presented in [38] was employed to handle network slice configurations of large dimensions in terms of actions without compromising resources. In terms of DRL performance, pointer networks (Ptr-Net) [39] are useful for combinatorial trajectory planning. Additionally, [40], [41], [42] used an actor–critic Ptr-Net hybrid architecture to achieve the fair coexistence between heterogeneous nodes in unlicensed bands. These compelling findings provide a foundation for our current approach.

III. PROPOSED FRAMEWORK

To evaluate the effectiveness of the suggested DRL solution for the GMEVTP problem, a mathematical representation of

TABLE 1. MIPOPT scenario known information.

Info	Description
$\mathcal{C}$	Overall set of clusters, where $c \in \mathcal{C}$ denotes one cluster.
$\mathcal{E}$	The set of clusters with edge devices inside their transmission range, $\mathcal{E} \subseteq \mathcal{C}$.
$\mathcal{I}$	Overall set of IoT devices, where $i \in \mathcal{I}$ denotes a specific IoT device.
$\mathcal{I}(c)$	The set of IoT devices at the transmission range of the center of cluster $c \in \mathcal{C}$.
$v(c,i)$	This flag identifies whether device $i \in \mathcal{I}$ is covered by $c \in \mathcal{C}$.
$t(c_j, c_k)$	The travel time between the centers of the clusters c_j and c_k, where $c_j, c_k \in \mathcal{C}$.
$pk(i)$	The amount of packets to be transmitted by IoT device $i \in \mathcal{I}$.
$sz(i)$	The average packet size (bits) transmitted by device $i \in \mathcal{I}$.
$TTL(i)$	The time to live for packets from device $i \in \mathcal{I}$. After that, data are considered expired.
$ToA(i,c)$	This represents the airtime for one bit when transmitting a packet from/to IoT device $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
$PRR(i,c)$	The packet reception ratio when transmitting a packet from/to $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
$NoC(i,c)$	This is the non-collision probability when transmitting a packet from/to $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
τ^{MAX}	The time limit for data collection.

the issue is created, and a maximum limit is determined. The DRL model results are then compared to such maximum limits. Therefore, the solution framework is composed of two primary elements: The MIPOPT mathematical model and the A2PC DRL agent.

The goal of MIPOPT is to determine the maximum amount of data that may be collected and delivered within a certain time frame. As a convention, this data which is obtained from LoRaWAN nodes and delivered to the edge is going to be referred to as *processed data* throughout this article.

Concisely put, the A2PC model was built atop a successful Ptr-Net architecture [40], and the agent was trained using the A2C algorithm [43]. To incorporate GMEVTP requirements (i.e., location and visit time predictions), our A2PC implementation uses two heads and is trained with an augmented actor-critic loss function.

TABLE 2. MIPOPT decision variables.

Variable	Description
λ_{c_j,c_k}	This takes the value 1 if the gateway travels from the center of cluster $c_j \in \mathcal{C}$ to the center of cluster $c_k \in \mathcal{C}$; it is 0 otherwise.
τ_c^{A}	This denotes the arrival time of the gateway at the center of cluster $c \in \mathcal{C}$.
τ_c^{D}	Following the above, this is the departure time of the gateway from the center of cluster $c \in \mathcal{C}$.
τ_c^{T}	When visiting cluster $c \in \mathcal{C}$, this is the start of the gateway's reception/transmission time window .
μ_c	This is the period spent transmitting data. For simplicity, inside the MIPOPT, this is referred as to as the *service time* of the gateway at cluster $c \in \mathcal{C}$.
ρ_c^i	This represents the percentage of packets of IoT device $i \in \mathcal{I}(c)$ collected by the gateway when visiting cluster $c \in \mathcal{C}$.
φ_{c_j,c_k}^i	This represents the percentage of packets from IoT device $i \in \mathcal{I}$ collected by the gateway when visiting cluster $c_j \in \mathcal{C}$, but also delivered in $c_k \in \mathcal{E}$.
β_{c_j,c_k}^i	This flag is set to 1 if $c_k \in \mathcal{E}$ is serving as a delivery cluster for the data of IoT device $i \in \mathcal{I}(c_j)$, $c_j \in \mathcal{C}$.

The next sections provide a detailed explanation of both of these framework components.

A. UPPER BOUND

The following MIPOPT model provides an upper bound for the amount of data that can be processed within a specified timeframe and is used as a reference when analyzing DRL agent performance. The model presupposes the existence of cluster coverage zones and assumes that gateways traverse these zones to gather and transmit data. Consequently, the cluster regions encompass IoT devices and edge systems. The scenario known information is summarized in Table 1. Note that $PRR(i,c)$ and $NoC(i,c)$ are based on the average packet size associated with a particular spreading factor and code rate (SF-CR) [44], [45]. These are predetermined based on the distance between IoT device $i \in \mathcal{I}$ and the center of cluster $c \in \mathcal{C}$. Following [44], [45], the conventional ALOHA protocol was employed to calculate the number of collisions, $NoC(i,c)$. The Table 2 presents the decision variables of the the model. In order for the gateway to start or end the journey in any cluster, two virtual clusters are incorporated into $\mathcal{C}$: *i*) c_1, which contains only outgoing edges, and $t(.) = 0$ for

TABLE 3. MIPOPT at a glance.

Component	Description
Gateway's journey	Flow conservation law to ensure no gaps from the beginning until the end of the gateway journey.
Data collection	Gathering of data from devices during cluster traversal.
Data delivery	Complete delivery of collected data only to edge devices.
Data expiration	Delivery of data to the edge before it becomes outdated.
Arrival, departure, and transmission	Setting up time windows at each stage of the journey.
Visit time	Condition visit times according to the packet reception ratio and the non-collision probability.
Final arrival time	Final arrival time cut-off.
Non-negative assignment	Type and bounds for variables.
Goal	Maximizing the amount of processed data.

each outgoing edge; *ii*) $c_{|\mathcal{C}|}$, which has only incoming edges; and $t(.) = 0$ for each incoming edge. These virtual clusters do not encompass devices and are linked to each non-virtual cluster to serve as the initial and final points for gateway trajectories.

The model constraints are discussed next, and summarized in Table 3.

1) CONSTRAINTS

This model defines several requirements that must be fulfilled during solution calculation. They are described as follows.

– Gateway's journey:

The gateway must commence its journey at virtual cluster c_1 and conclude at virtual cluster $c_{|\mathcal{C}|}$. As previously stated, transitioning from c_1 to any non-virtual cluster and from any non-virtual cluster to $c_{|\mathcal{C}|}$ incurs no costs. Consequently, it is guaranteed that a gateway can initiate or conclude its route in any non-virtual cluster. This is accomplished using the following equation:

$$\sum_{\{c_k \in \mathcal{C}\}} \lambda_{c_j,c_k} - \sum_{\{c_k \in \mathcal{C}\}} \lambda_{c_k,c_j} = \begin{cases} 1, & \text{if } c_j = c_1 \\ -1, & \text{if } c_j = c_{|\mathcal{C}|} \\ 0, & \text{otherwise} \end{cases}, \forall c_j \in \mathcal{C} \tag{1}$$

Constraint (1) enforces the flow conservation principle on the gateway path. This law ensures that there are no interruptions along the paths from c_1 to $c_{|\mathcal{C}|}$.

– Data collection:

$$\rho^i_{c_j} \leq v(c_j, i) \times \sum_{\{c_k \in \mathcal{C}\}} \lambda_{c_j,c_k}, \forall c_j \in \mathcal{C} : \mathcal{I}(c_j) \neq \emptyset, \forall i \in \mathcal{I} \tag{2}$$

$$\sum_{\{c \in \mathcal{C}: i \in \mathcal{I}(c)\}} \rho^i_c \leq 1, \forall i \in \mathcal{I} \tag{3}$$

Constraint (2) guarantees that data are gathered exclusively from clusters falling within the transmission range of the IoT device and are part of the gateway's path. Constraint (3) specifies that data can only be collected when the gateway is present in any of the clusters that cover it.

– Data delivery:

$$\sum_{\{c_k \in \mathcal{E}\}} \varphi^i_{c_j,c_k} = \rho^i_{c_j}, \forall c_j \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{4}$$

$$\sum_{\{c_j \in \mathcal{C}\}} \sum_{\{i \in \mathcal{I}(c_j)\}} \varphi^i_{c_j,c_k} \leq \sum_{\{c_j \in \mathcal{C}\}} \lambda_{c_j,c_k} \times |\mathcal{I}| \times |\mathcal{C}|, \forall c_k \in \mathcal{E} \tag{5}$$

Constraint (4) guarantees the delivery of all collected data to the edge, whereas Constraint (5) ensures that data are delivered specifically to edge locations that are part of the gateway's path.

– Data expiration:

When the gateway receives data from an IoT device, it is necessary to ensure that it does not expire until delivery to the edge storage system. The following restrictions ensure this condition. The objective function guarantees that the data are always as fresh as possible by minimizing the time required for gateways to collect and send data, assuming that lifetime constraints are met.

$$\beta^i_{c_j,c_k} \geq \varphi^i_{c_j,c_k}, \forall c_j, c_k \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{6}$$

$$\tau^{\mathrm{T}}_{c_k} - \tau^{\mathrm{T}}_{c_j} \leq TTL(i) + \Delta \times (1 - \beta^i_{c_j,c_k}), \forall c_j, c_k \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{7}$$

where Δ denotes large values. Constraint (6) determines the binary information, noting if cluster c_k supports the delivery of data from $i \in \mathcal{I}(c_j)$, Constraint (7) ensures that data are delivered to the edge before becoming outdated.

– Arrival, departure and transmission times:

To ensure that the gateway remains in the cluster for the duration needed to collect or deliver the specified data, it is crucial to determine both the arrival and departure times of the gateway at the center of the cluster. The following conditions must be considered:

$$\tau^{\mathrm{A}}_c \leq \tau^{\mathrm{T}}_c \leq \tau^{\mathrm{D}}_c, \forall c \in \mathcal{C} \tag{8}$$

$$\tau^{\mathrm{T}}_c + \mu_c \leq \tau^{\mathrm{D}}_c, \forall c \in \mathcal{C} \tag{9}$$

$$\tau^{\mathrm{D}}_{c_j} + St(c_j, c_k) - \Delta \times (1 - \lambda_{c_j,c_k}) \leq \tau^{\mathrm{A}}_{c_k}, \forall c_j, c_k \in \mathcal{C} \tag{10}$$

where Δ denotes large values. Constraint (8) enforces the inclusion of transmission time windows within the arrival–departure period, whereas Constraint (9) incorporates the service time within it. Constraint (10) requires that the gateway is not allowed to reach the center of cluster c_k before $\tau^{\mathrm{D}}_{c_j} + t(c_j, c_k)$ when traveling from c_j to c_k. The utilization of Δ enables the fulfillment of the constraints based on the determined value of λ_{c_j,c_k}.

– Visit time:

The duration of a gateway remaining in a cluster must be equal to or greater than the time required to collect and/or deliver the required data. This duration increases when the value of $PRR(.)$ or $NoC(.)$ decreases [44].

$$\mu_{c_j} \geq \sum_{\{i \in \mathcal{I}(c_j)\}} \frac{pk(i) \times sz(i) \times ToA(i, c_j)}{PRR(i, c_j) \times NoC(i, c_j)} \times [\rho^i_{c_j} + \sum_{\{c_k \in \mathcal{C}\}} \varphi^i_{c_k,c_j}], \forall c_j \in \mathcal{C} \tag{11}$$

Constraint (11) includes the spectrum utilization time for both data collection and delivery when determining the service time of the gateway in a certain cluster.

– Final arrival time:

A subsequent constraint is necessary to restrict the final arrival time (i.e., temporal window) for data gathering.

$$\tau^{\mathrm{A}}_{c_{|\mathcal{C}|}} \leq \tau^{\mathrm{MAX}} \tag{12}$$

– Non-negative assignment to variables:

$$\lambda_{c_j,c_k}, \beta^d_{c_j,c_k} \in \{0, 1\}; \quad \tau^{\mathrm{A}}_c, \tau^{\mathrm{D}}_c, \tau^{\mathrm{T}}_c, \mu_c \geq 0; 0 \leq \rho^i_c, \varphi^i_{c_j,c_k} \leq 1 \tag{13}$$

2) OBJECTIVE FUNCTION

Finally, it comes the definition of the objective function of the MIPOPT, expressed as the maximization of the processed data for all nodes in all clusters.

$$\text{Maximize} \sum_{\{i \in \mathcal{I}\}} \sum_{\{c \in \mathcal{C}: i \in \mathcal{I}(c)\}} \rho^i_c \tag{14}$$

Several critical points must be emphasized here. Reducing travel and visit times leads to shortened data processing, resulting in energy conservation and improved data freshness. A crucial assumption underlying this requirement is that the total energy of a traveling gateway is inexhaustible. The SF setting under the most unfavorable conditions combined with the greatest distance paths traversed by the gateway are utilized within the maximum time for simulations, including traveling between clusters. This specification ensures a well-defined τ^{MAX} that does not exhaust the battery of the gateway until its journey is complete. Hence, the objective function encourages energy savings. Furthermore, the implementation of time non-violation in Constraints (7) guarantees the fulfillment of stringent data freshness requirements.

B. MIPOPT COMPLEXITY ANALYSIS

Such MIPOPT model is more complex than the vehicle routing problem with time windows addressed in [46] because collect/drop places are also being decided, while a time window for data collection is imposed. Vehicle routing and scheduling problems are known to be NP-Hard, meaning that the discussed MIPOPT problem will also be hard. Solvers will only be able to find solutions for small instances of the problem [13].

Besides the just mentioned complexity, such kind of models require reality assumptions or simplifications so that building constraints becomes feasible. For this reason, the MIPOPT is used in this article as an upper bound, and DRL techniques are used to develop an agent able to incorporate any underlying dynamics into its decisions and, after trained, make decisions in real time [44], [45].

C. A2PC MODEL

With DRL, an agent learns to interact with the environment by observing and reacting to the action of other agents and the features in the environment. Accordingly, as the environmental state changes, the agent receives rewards based on outcomes. Given a predefined reward system, the agent's objective is to accumulate the highest possible outcome, G_t, over all interactions

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_0 = s_t\right], \tag{15}$$

where $r(s_t, a_t)$ is the reward for taking action a_t in state s_t at time t. The leading coefficient, γ, is a discount factor that balances the short- and long-term rewards.

For the agent to interact efficiently with the environment, it must follow a policy π to determine the action to be performed in a given state. A policy is optimal, π^*, when following it produces the maximum G_t.

There are two primary ways to learn a policy: value-based and policy-gradient methods. The current study leverages policy-gradient techniques.

In value-based learning, a policy is learned from the value function, $V(s)$, which measures how good it is to be in in state s.

$$V(s) = \mathbb{E}\left[G_t | s_t = s\right] \tag{16}$$

To measure the goodness of taking action a at state s, the quality (Q) function is used:

$$Q(s, a) = \mathbb{E}\left[G_t | s_t = s, a_t = a\right] \tag{17}$$

Furthermore, to obtain the relative importance of action a and determine if choosing it is better than the average performance of the policy, the advantage function is used:

$$A(s, a) = Q(s, a) - V(s), \tag{18}$$

which can be approximated as

$$A(s_t, a_t) = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t) \tag{19}$$

With policy-gradient learning, a policy is represented as a parametric function for which the objective is to learn the set of parameters that maximizes the outcome. Let Λ be a trajectory (i.e., sequence of transitions), such that $\Lambda = s_0, a_0, \ldots, s_{T-1}, a_{T-1}$ for an episode of length T. Using the probability general product rule, π can be defined as

$$\pi(\Lambda;\theta) = p(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t;\theta)p(s_{t+1}|s_t,a_t), \tag{20}$$

where p is a predefined transition probability function, and θ is the set of unknown parameters that can be estimated by maximizing objective function J.

$$\theta^* = \arg\max J(\theta), \tag{21}$$

where J can be formulated as

$$J(\theta) = E_{\Lambda\sim\pi(\Lambda;\theta)}r(\Lambda) = \int \pi(\Lambda;\theta)r(\Lambda) \tag{22}$$

Following the logarithmic differentiation, there is

$$\nabla_\theta J(\theta) = E_{\Lambda\sim\pi(\Lambda;\theta)}\left[\sum_{t=0}^{T-1}\nabla\log(\pi(a_t|s_t;\theta))G_t\right] \tag{23}$$

By applying the gradient ascent rule with a predefined learning rate, α, there is

$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\theta) \tag{24}$$

These three steps of sampling a trajectory $\Lambda \sim \pi(\Lambda;\theta)$, (23), and (24) are known as the "REINFORCE" algorithm, which is used to train the model in this study. Considering these bases, the proposed model is described as follows.

1) STATE

State s_t at time t is a vector with the following five features:

$$s_t = [s_t^{\mathrm{P}}, t, s_t^{\mathrm{B}}, TTE, s_t^{\mathrm{RB}}], \tag{25}$$

where

- s_t^{P} is the current position of the gateway.
- t is the elapsed time thus far.
- s_t^{B} is the amount of processed data up to this point in time.
- TTE is the time to expire of the data in transit.
- s_t^{RB} is the amount of the remaining data to be processed.

2) ACTION SPACE

This study focuses on two heterogeneous action spaces. The first specifies where the gateway moves (location), and the second represents the visit time associated with the selected move.

Notably, during problem formulation, LoRaWAN devices are grouped into clusters according to their spatial distributions. Hence, the first action corresponds to the cluster number. The location-related action space is given by

$$\mathcal{A}^{\mathrm{C}} = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}, \tag{26}$$

where c_1 and $c_{|\mathcal{C}|}$ are two virtual clusters included in the mathematical formulation.

The second action (i.e., visit time) is mapped into *timesteps* by considering real-world scenarios [12], where the upper limit of the circuit trajectory reaches 10,800s. Timestep conversion is performed by stipulating units of 5s each. Hence, each timestep is equivalent to 5s and is valid because it accommodates up to 51 bytes of transmission and reception. These values represent the largest messages in the worst case LoRa scenario [47]. However, following real-world scales and the given considerations, this action space is defined as:

$$A^{\mathrm{T}} = \{1, 20, 40, 60, 80, 100, 120, 140, 160, 180\}, \tag{27}$$

where units are measured in timesteps. Visit times greater than 180 timesteps fall outside realistic range and time periods.

3) REWARD

Developing a reward function for A2PC is challenging as there are two action spaces (i.e., location and visit time). Thus, the proposed reward must be proportional to the number of bytes collected or delivered to enable the A2PC agent to filter non-productive moves.

$$r(s_t, s_t^{\mathrm{P}}, v_t) = \begin{cases} s_t^{\mathrm{B}} - s_{t-1}^{\mathrm{B}}, & \text{if} \quad s_t^{\mathrm{P}} \notin \mathcal{E}, \text{ and byte collected} \\ s_t^{\mathrm{B}}/v_t, & \text{if} \quad s_t^{\mathrm{P}} \in \mathcal{E}, \text{ and } TTE > 0 \end{cases} \tag{28}$$

where TTE (time to expire) is applied to all messages issued by the LoRaWAN nodes and resets whenever the gateway visits the edge.

D. A2PC ARCHITECTURE

The action space can be a bottleneck in developing and deploying DRL solutions. This usually happens because agents hardcode the dimensions of the action space. Hence, any changes in the action space are difficult to incorporate without retraining (which are demanding in most use cases).

Another issue is the complexity added when an additional decision is required. This is true in this case, where not only the location of the gateway but also the visit time must be determined. In this situation, there are two fundamental challenges. First, the difficulty of learning a high-dimensional action space. Second, the determination of a suitable architecture and loss function.

To address these challenges, we developed an augmented pointer–critic architecture and improved the A2C method to incorporate it. The Ptr-Net architecture is leveraged to represent variable-length dictionaries and is applied to solve combinatorial problems [36], [40], [48]. In a typical Ptr-Net [49], [50], the input sequence of elements is encoded into a representation space where the decoder generates pointers (indexes) to input one pointer at a time. These pointers select available elements from the input sequence until a predefined condition is satisfied (end of the sequence token is generated). Furthermore, the masked attention mechanism is used to

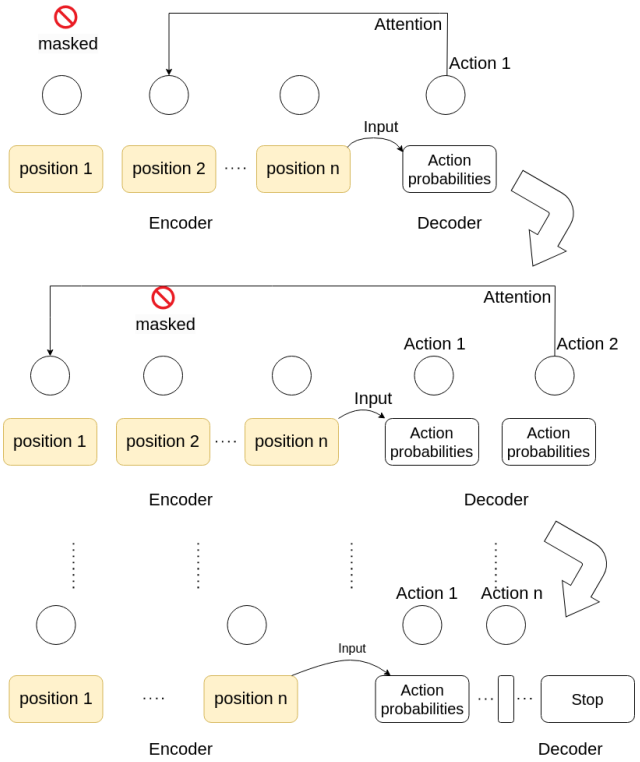

FIGURE 2. Masked attention concept of the pointer network.

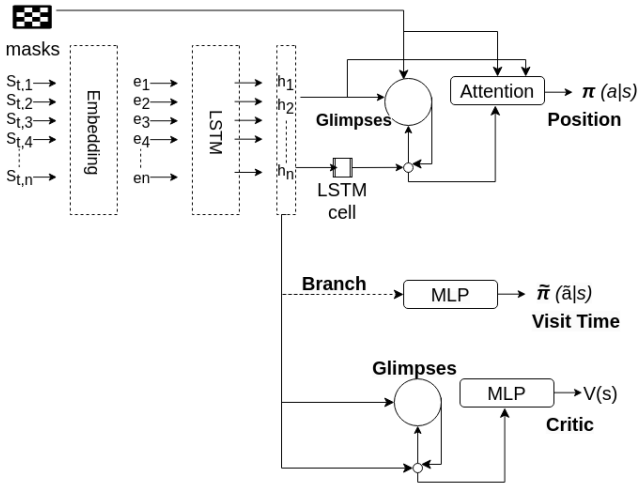

FIGURE 3. Action-branching high-level diagram.

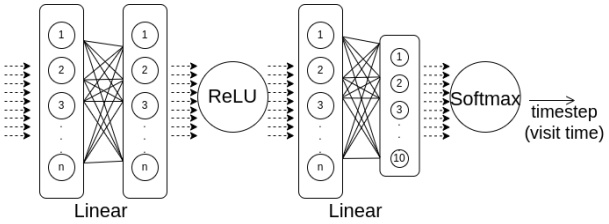

FIGURE 4. Multilayer perceptron visit time branch diagram.

avoid non-repetitive actions, which makes sense in this work as the gateway is intended to move to different positions at each timestep. The entire process, as depicted in Figure 2, comprises pointers that are filtered by masking mechanisms and select feasible tokens present in each element of the output sequence.

E. ACTION-BRANCHING (MULTI-HEAD PTR-NET)

The gateway travel-planning problem revolves around the selection of positions and visit times. The dimensions of the action space grow rapidly, depending on the scenario. The combination of moves and visit timesteps can not be easily achieved with a single output as they operate in fundamentally two different spaces. Therefore, two heads (i.e., location and visit time) are used as shown in Figure 3. The location is selected using a pointer head that aims at the position index in the input state. The visit time decision is made by another head.

In this scheme, a single baseline critic evaluates policies that guide both actors (heads). This method allows the critic to capture the behavior of both actors and offers advantages as it renders the model simple, efficient, and easy to extend. This keeps the critic's focus on a single objective and enables it to estimate the value function that can be used for both actors' losses.

As shown in Figure 4, the visit time head is a multilayer perceptron (MLP) that directly branches from the shared LSTM of the model. Rather than using a full copy of the location head actor, this straightforward design has three advantages. First, visit time sequences do not benefit from action masking as they may have repeated value sequences. Second, the visit time sequence does not benefit from the pointer head as their prediction sequences do not point back to the input. Third, as a design choice, the visit time head is lightweight and does not overcomplicate the existing Ptr-Net model.

F. LOSS FUNCTIONS

For this augmented actor-critic, the loss function takes into account both heads' losses and is formulated as follows:

- actor position loss:

$$-\sum_{t=0}^{T-1} log(\pi(a_t|s_t:\theta))A(s_t,a_t) \tag{29}$$

- actor visit time loss:

$$-\sum_{t=0}^{T-1} log(\tilde{\pi}(\tilde{a}_t|s_t:\theta))A(s_t,\tilde{a}_t), \tag{30}$$

  where $\tilde{\pi}$ is estimated from the visit time actor.

- critic loss:

$$\frac{1}{2}\sum_{t=0}^{T-1} A(s_t,a_t)^2 + \frac{1}{2}\sum_{t=0}^{T-1} A(s_t,\tilde{a}_t)^2 \tag{31}$$

input: T **= episode length,** α_{entropy} **= entropy factor**
output: Trained A2PC model

Initialize actors π_θ, $\tilde{\pi_\psi}$ network parameters;
Initialize critic V_ϕ network parameters;
Initialize state s_0;

//sample the entire episode returning the
//list of state, actions, and reward
$s, a, \tilde{a}, r \leftarrow SampleEpisode(s_0, V_\phi, \pi_\theta, \tilde{\pi}_\psi, T)$;

$E \leftarrow ComputeEntropy(\pi_\theta, \tilde{\pi}_\psi)$;

$g_{\theta,\psi} \leftarrow \sum_{t=0}^{\mathrm{T}-1} A(s_t, a_t)\nabla_\theta log(\pi(a_t|s_t)) + A(s_t, \tilde{a}_t)\nabla_\psi log(\tilde{\pi}(\tilde{a}_t|s_t)) + \alpha_{\text{entropy}} E$;
$L_\phi \leftarrow \frac{1}{2}\sum_{t=0}^{\mathrm{T}-1} A(s_t, a_t)^2 + \frac{1}{2}\sum_{t=0}^{\mathrm{T}-1} A(s_t, \tilde{a}_t)^2$;

$\theta, \psi, \phi \leftarrow ADAM(\{\theta, \psi, \phi\}, (g_{\theta,\psi}, \nabla_\phi L))$;

Return trained actors (location and time)
neural networks.

Algorithm 1 Training Phase of the A2PC Agent

- entropy loss:

$$-\{\sum_{t=0}^{T-1}\pi(a_t|s_t)log(\pi(a_t|s_t)) + \sum_{t=0}^{T-1}\tilde{\pi}(\tilde{a}_t|s_t)log(\tilde{\pi}(\tilde{a}_t|s_t))\}, \quad (32)$$

where $a \in \mathcal{A}^{\mathrm{C}}$ and $\tilde{a} \in \mathcal{A}^{\mathrm{T}}$

G. A2PC TRAINING

The proposed model is trained using the A2PC algorithm, as shown in Algorithm 1.

H. A2PC ANALYTICAL PROPERTIES

To better understand the proposed A2PC model, its complexity and convergence, it is natural to analyze its components: the RL components, the neural network architecture, the loss function, and the training algorithm.

The proposed A2PC inherits the main RL properties from the A2C agent. The actor and critic of the A2C are also in the A2PC. To model the multi-action requirements of the GMEVTP problem, A2PC has two actors for the location and visit time predictions. To show that these two actors are adequately modeled in the A2PC, they are tracked in the proposed neural network, the loss function, and the training algorithm.

The proposed neural network of the A2PC has three heads. Two for the actors and one for the critic. All heads share the same hidden space modeled by the LSTM layer (Figure 3). However, the extension for the visit time prediction actor head is a simple (only two layers) projection followed by non-linearity, which makes it resilient to gradient issues. Therefore, this new head does not modify the representation space consumed by the location actor or the critic.

On the same note, the loss of the visit time prediction head is modeled the same way as the location prediction (29) and (30). In other words, each head learns a probability distribution over each head's action space and shares the same critic. Since the critic now moderates two actors, its loss (31) is based on the average of the advantage function of both types of actions. The same simple combination is also used when calculating the average entropy of both action policies (32). It is clear that this linear combination, in the advantage and entropy losses, extends the loss function naturally and does not affect the convergence properties of the REINFORCE algorithm.

With these architectural and loss function properties, the training algorithm maintains the A2C training properties since all loss terms are differentiable, and no gradient (or convergence) issues need to be dealt with from a simple projection of the state representation space. This was confirmed when implementing the A2PC agent, where no convergence issues were found.

Regarding the complexity of proposed model and algorithm, it is possible to consider a single forward pass of the model given the online settings of the trained actor models are the only deployed components. The complexity of the model depends mainly on the complexity of the encoding layer, which is represented by the LSTM module. This layer is the bottleneck and requires the highest computational demands in the entire model. However, the complexity of the LSTM is fixed on each step [51], and the complexity of the encoder is the sum of the complexity of the embedding layer and the LSTM, $\mathcal{O}((k_1 + 2k_2)|\mathcal{A}|)$ where $k_1 = 5$ is the feature space dimension and $k_2 = 128$ is the embedding space dimension, and $|\mathcal{A}| = 16$ is the action space dimension. On the other hand, the decoder and actor complexity, is dominated by the attention complexity, which is $\mathcal{O}(k_2)$). Therefore, the overall complexity is $\mathcal{O}((k_1 + 2k_2)|\mathcal{A}|) + O(k_2)$.

IV. PERFORMANCE EVALUATION

This section provides a comprehensive overview of the proposed pipeline, and then discusses the evaluation approach, KPIs, scenario preparation, convergence checking, implementation, and results.

A. PIPELINE DESIGN

To establish a benchmark between the A2PC and the optimal model, a new evaluation pipeline based on the NS3-LoRa simulator, CPLEX, and OpenAI Gym is developed.

The workflow and its source code is publicly available [52]. Its four main steps can be summarized as follows:

- Step 1 (Data collection): Here, the NS3-LoRa simulator from [53] generates node deployment scenarios, defines data transmissions, and runs an RL agent that optimizes LoRa communication settings. In brief, Step 1 defines the LoRaWAN clusters' topology, including distances,

way-point positions, device characteristics, data to be processed, and arrival times. During these procedures, the energy savings and optimal transmissions are ensured by the agent introduced in [8].

- Step 2 (Upper bound selection): Here, the output of Step 1 is used to parameterize mathematical optimization. This procedure produces the main KPI of the system (i.e., total data processed in bytes) by the traveling gateway. That is, the mathematical optimization process is executed such that the MIPOPT from Subsection III-A is solved with the CPLEX optimizer to establish an upper bound.
- Step 3 (A2PC deployment): In this step, the A2PC agent is employed to learn the locations and visit times for the mobile gateway under the same setup used in Step 2.
- Step 4 (Analytics): A comparative analysis of the results of both MIPOPT and A2PC solutions is conducted. Apart from comparing the main KPI, an additional evaluation is performed for the A2PC agent capabilities. Successfully trained models are tested against scaled clusters while retaining the same topology by randomly cloning or subtracting end nodes inside clusters. Expanding the scenarios with replica nodes enables the assessment of whether the A2PC has delivered scalability capacity to the agent.

Additionally, the above steps can be better visualized by the diagram representation in the Figure 5.

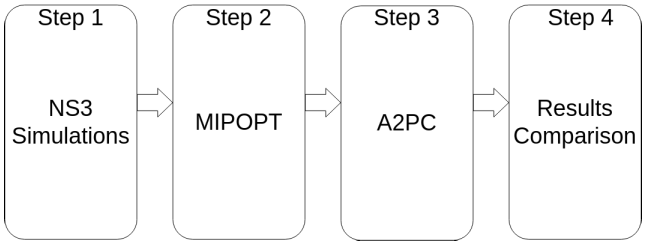

FIGURE 5. Evaluation pipeline high overview diagram.

1) KPIS

There are two KPIs. The first is the amount of data processed within an arbitrary travel duration, $\mathcal{T}^{\text{MAX}}$, and the other is the scalability of the DRL solution.

For the first, an arbitrary travel duration is obtained from Step 1, which is assumed to be half the most extended period for processing all data in all clusters. This predefined period ensures that the optimization upper bound never reaches 100%, even with all the packets generated by the data collection simulations. Moreover, if long-lasting periods were to be used, the A2PC agent and the MIPOPT would process nearly all the data available, thus defeating the idea of comparing one to another. The constrained time forces the A2PC agent and MIPOPT to determine the maximum amount of data to be processed within an arbitrary timeframe. This establishes a tangible limit that enables a comparison between the two and follows empirical deployment requirements.

The solution's scalability is the second KPI. This KPI has rarely been addressed in the literature; however, it is used here to evaluate the capability of A2PC to perform while varying the number of nodes, but keeping the same network topology.

Apart from the KPIs, two other metrics have significant weight when assessing the solution. First, it comes the visit time learned by the A2PC, as this pattern helps in understanding the demands of different scenarios. Given its influence, this metric is also discussed in the results analysis. Secondly, a comparison between the agent GMwES and A2PC is also essential. Since GMwES was proved successful in [12], such a comparison stands valuable and is conducted in Section IV-B.4.

2) SCENARIO SETUP

The standards implemented in the LoRaWAN-NS3 simulator accurately model real-world systems [54]; thus, they are also used in the experiments. The assessment of the A2PC is done by varying the conditions, which are, in turn, associated with the scenario type they are in. Categorization of the scenarios takes into account two main factors. First, there is the node density distribution, where scenarios can be classified as dense or sparse. Secondly, scenarios can be large and small. Altogether, there are four possible combinations, and their conventions are better visualized in Table 4. It is noteworthy to highlight the type of cluster proportions, which uses a realistic ratio of 25% of edge clusters concerning the number of total clusters

TABLE 4. Deployment scenarios.

Scenario	No. Clusters	Avg No. Devices per Cluster
Small & Sparse (SS)	8	1 to 7
Small & Dense (SD)	8	1 to 7
Large & Sparse (LS)	16	8 to 16
Large & Dense (LD)	16	8 to 16

The moving gateway is assumed to maintain a constant speed of 60 km/h [55], and the variation in visit times compensates for the need for any delays or accelerations during the journey. The average distance between any two clusters is 5.8 km. In large scenarios, the maximum duration of the gateway journey could reach 5,400s, whereas in small scenarios, it could reach 3,000s. Figure 6 presents a sample SD scenario, which provides a more comprehensive illustration of how variations in size and density are characterized in the experiments. The dots represent the LoRaWAN nodes, whereas the circles represent the respective clusters.

The volume of data, as well as the average sizes of messages are based on real-world examples [56], [57], [58], and following that, Table 5 summarizes the parameters and conventions used in the experiments.

3) IMPLEMENTATION DETAILS

The A2PC was implemented in Python 3.8, PyTorch 1.10 [52], and Open-AI Gym v0.26.0. The model was run on a

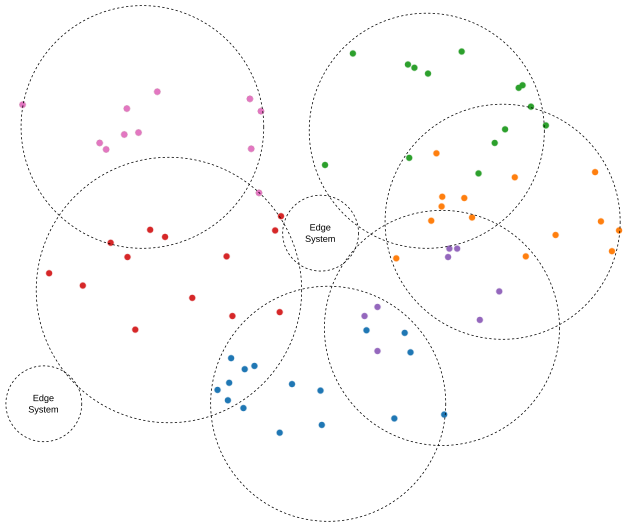

FIGURE 6. **Small dense scenario sample.**

TABLE 5. **Experiments setup information.**

Information	Value
Scenarios*	SS, SD, LS and LD
Clusters with edge nodes	25% of all clusters
TTE	1,200s
Message avg size	18 bytes
No. messages per node	1–5
$\mathcal{T}^{\text{MAX}}$	Half of Step 1 simulation time
A2PC discount factor γ	0.9
A2PC learning rate α	0.001
A2PC entropy factor ϵ	0.001
A2PC training episodes	7,000 (small) & 14,000 (large)

Ubuntu 20.02 host system comprising an 8 vCPU Intel i7, 16-GB RAM, and Nvidia GeForce RTX 2080 GPU GV102 with 11,264 MB.

Finally, the A2PC parameterization followed the assumptions described in Sections IV-A.4 and III-E. The hyperparameters were also fine-tuned following similar configurations that offered good functionality in previous experiments [12], and these values also appear compiled in Table 5.

4) A2PC CONVERGENCE

Convergence analysis may be conducted by considering the type of scenario and the average accumulated rewards. Figure 7 displays the episode count at which the reward is stable. Each scenario type was subjected to a preliminary set of 20 exploratory trials, and the results show a common convergence pattern by scenario sizes. Small scenarios tend to converge at 7k, while large converge around 14K episodes. Moreover, Figure 8 depicts the reward stability fitted curve over the number of episodes tried when training the agents. The illustration helps justify the choice of the parameters that appear in Table 5.

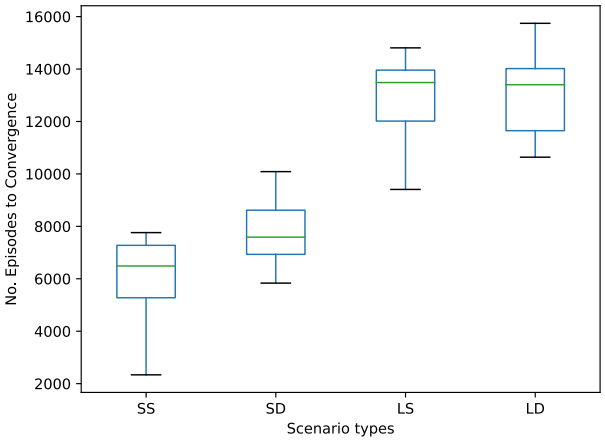

FIGURE 7. **Convergence of the agent in the different types of scenarios.**

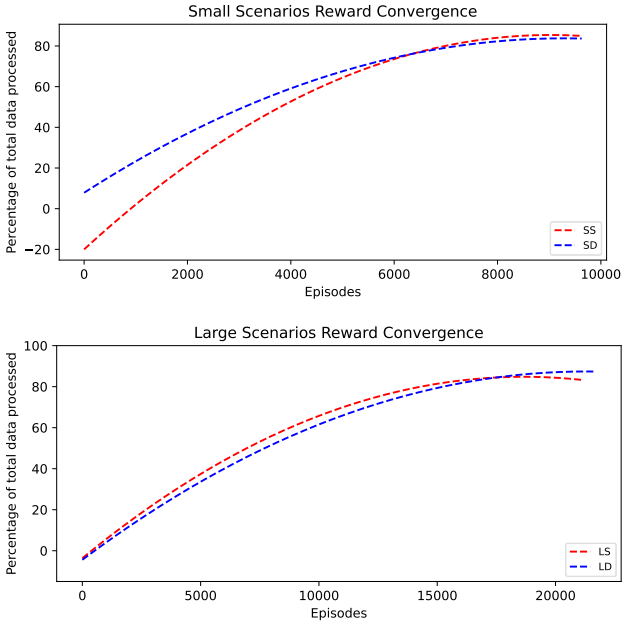

FIGURE 8. **Reward convergence curve in large and small scenarios.**

B. ANALYSIS OF RESULTS

In this subsection, the KPIs and metrics are evaluated based on the results of the experiments.

1) PROCESSED DATA

After training and executing 30 times per scenario, the results obtained by the A2PC are illustrated in Figures 9 and 10 for small and large scenario types, respectively.

The plots in these figures show the number of bytes processed by the model versus the optimal amount of data obtained using MIPOPT. Clearly, the A2PC performance maintained near-optimal results in all cases. Furthermore, in small scenarios, the A2PC dealt with fewer traffic overlaps, and the range of actions needed was smaller than in

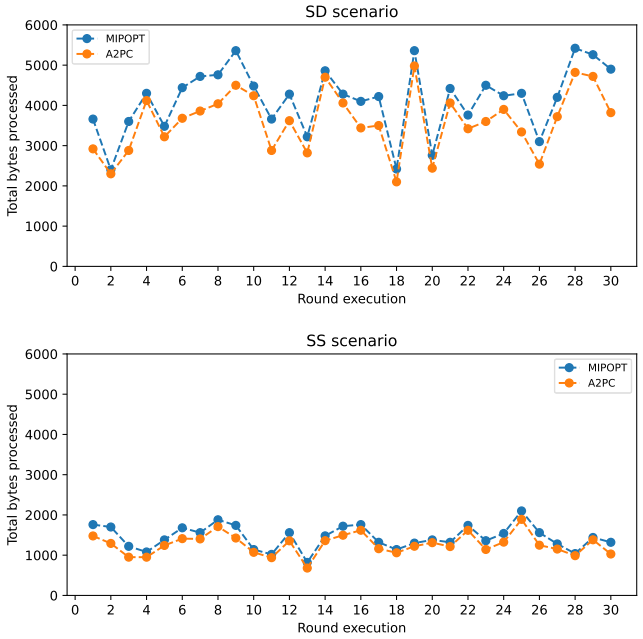

FIGURE 9. Bytes processed by A2PC vs. optimal values - Small scenarios.

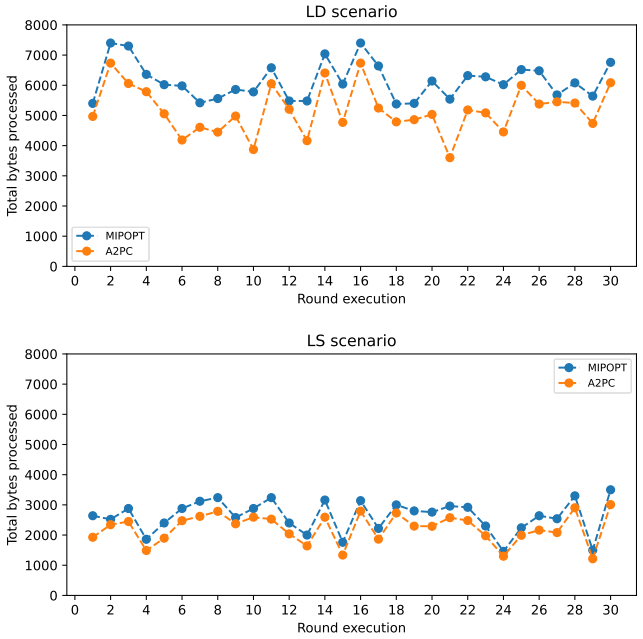

FIGURE 10. Bytes processed by A2PC vs. optimal values - Large scenarios.

large-cluster scenarios. Therefore, the performance was slightly better than that of the large scenarios.

When considering the total number of bytes processed in all rounds of execution, the analysis revealed that the A2PC system was unable to process 16,220 of the possible 124,460 bytes for the SD scenarios. Similarly, for the LD scenarios, 28,638 of 183,980 bytes could not be processed. These numbers support the idea that only a small portion of the available data remained unprocessed, even in situations with high data density. Detailed absolute comparisons are presented in Table 6.

TABLE 6. Total data processed for all A2PC executions.

Scenario	Total Bytes Available	Total Bytes Processed	Total Bytes Not Processed
SS	43,340	38,127	5,213
SD	124,460	108,240	16,220
LS	78,840	66,760	12,080
LD	183,980	155,342	28,638

The averaged results provided by A2PC compared with the upper bound are listed in Table 7, highlighting that A2PC is consistent in terms of sparsity. Moreover, a negligible drop can be noticed in dense scenarios, confirming A2PC efficiency regardless of scenario type.

TABLE 7. Avg. processed data achieved by the A2PC agent.

Scenario	Percentage of Total Data Processed
SS	85.29%
SD	84.35%
LS	81.77%
LD	81.55%

2) SCALABILITY

Additional experiments were conducted using successful DRL models from the previous training to test agent behavior under scaled-out and downscaled scenarios. The concept of expansion was based on cloning existing nodes and increasing the number of nodes inside the clusters. Hence, the agent was expected to anticipate the pattern of communication, thus performing as well as in the original environment.

The scalability tests were based on SS scenarios as this type produced the best overall results, as shown in Table 7. Thus, a random sample of 15 SS scenario type was selected from the executed rounds in IV-B to serve as the baseline. These scenarios were manipulated by introducing randomly cloned nodes, with the addition of 1, 5, and 10 units. The comparison between the baseline and scaled scenarios was performed based on the average performance differences among them, thus given in percentage. The reduced cluster sizes were tested using the same logic. A sample of 15 selected SS scenario types was subjected to random node removal performed in units of 1, 5, and 10. The execution of the same trained A2PC agents in the baseline and modified scenarios yielded results that can be compared to confirm the scalability capability of the Ptr-Net.

In Figure 11, the set of normalized cumulative distribution function (CDF) graphs depicts the performance divergence obtained by the execution of these agents. The values on individual lines denote the percentage difference of distributions in agent performance in the scaled scenario relative to the baseline scenario. Positive values indicate that the

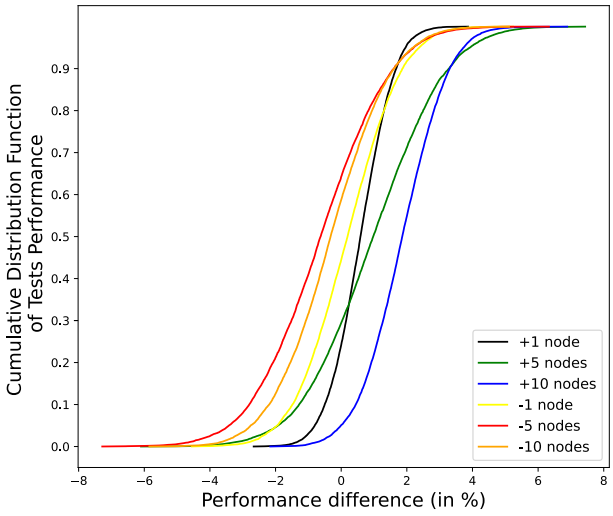

FIGURE 11. Variation of performance obtained by scaled scenarios in comparison with baseline scenarios.

agents outperformed the baseline scenarios, which were more prevalent in the scaled-out scenarios. However, the distribution began presenting negative incidences for scenarios in which the nodes were removed. This was more apparent in scenarios in which 5 or 10 nodes were subtracted. The CDF appears reasonable, as the A2PC agents may have visited clusters (based on previous learning) with absent nodes, thus decreasing overall performance.

Furthermore, it is possible to conclude that scaled-out environments were not an issue for the trained models. Given they preserve similar LoRa communication characteristics, agents can meet behavior patterns by collecting and delivering the same, and often slightly more, than the baseline scenarios.

3) VISIT TIME

The action branching method adds a singular characteristic to the A2PC agent. Hence, it is plausible that its usage and impact should be given more attention. Here, samples of the visit time actions learned by the A2PC during the 30 pipeline executions from Section IV-B.1 were compiled to provide an overview of their behavioral patterns. Figure 12 summarizes the number of occurrences for each timestep option, as in Definition (27), for all trained scenarios.

As a result, it is possible to distinguish between large and small scenarios as A2PC agents tend to concentrate their visit times in the lower timestep range for large scenarios while distributing them across the small ones. This outcome indicates that the A2PC can learn more rewarding behaviors according to the different topologies. Furthermore, the factor that most influenced this pattern was cluster size, not node density.

4) A2PC AND GMWES AGENTS COMPARISON

As a final analysis, a direct comparison between A2PC and GMwES [12] is proposed in this section. The scenarios and

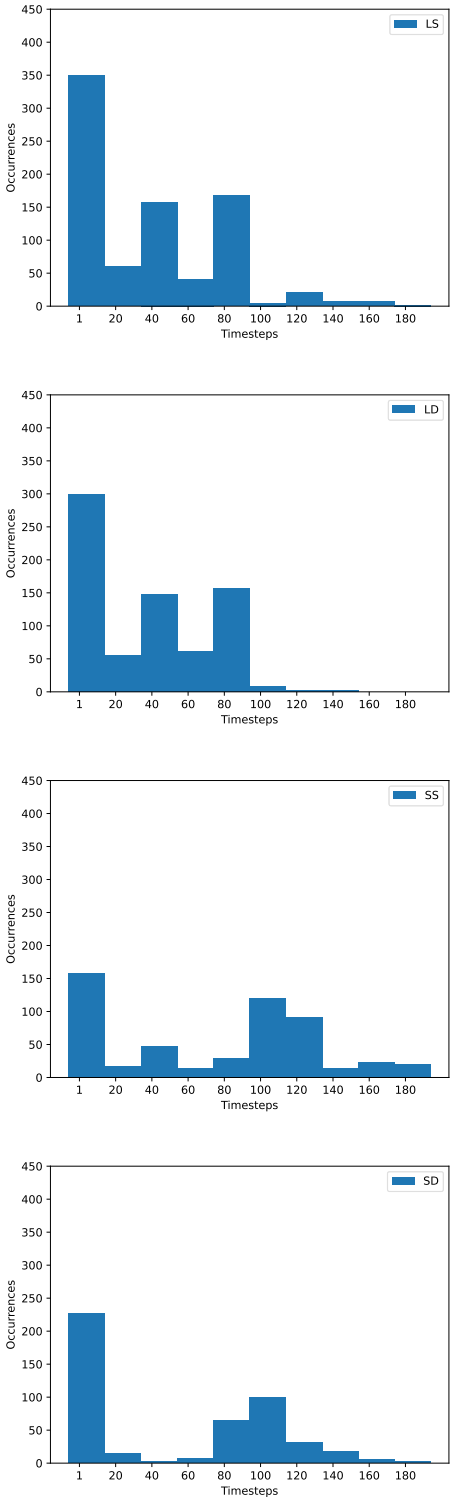

FIGURE 12. Timestep occurrences for large and small scenarios.

parameterization of the experiments follow the same specifications and conventions already described in Table 5. The main difference in here is that there is no optimal metrics, nor KPIs. This comparison is a straight verification between results of the 2 agents when executed under same scenarios. By implementing this approach, both agents would be

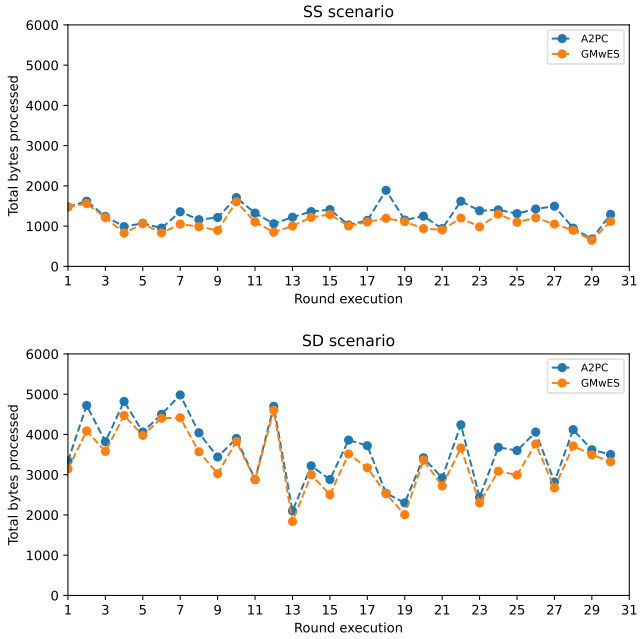

FIGURE 13. **Bytes processed per agent agent - Small scenarios.**

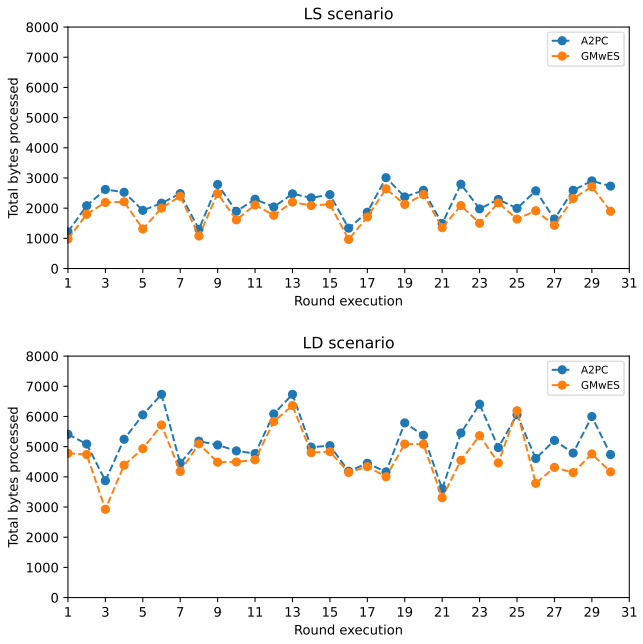

FIGURE 14. **Bytes processed per agent - Large scenarios.**

subjected to identical conditions and standards, thereby facilitating an equitable and impartial comparison.

Following the same approach as in Section IV-B.1, the experiments leverage NS3 and Python to produce configurations with a single gateway. Once the scenarios are configured, A2PC and GMwES agents are subject to training and tests sharing identical conditions at each execution fold. The performance comparison is done by checking the amount of bytes processed for each agent throughout a series of 30 rounds. Analogous to what is detailed in Section IV-A.1, the travel duration is half the maximum time achieved by the NS3 executions. The final results for these 30 executions are illustrated in Figure 13 and 14, for the small and large scenarios, respectively.

The figures provided demonstrate that, as measured by the number of bytes processed, A2PC maintains a consistently superior performance across all folds. This advantage becomes evident in Table 8.

The results corroborate the efficacy of the A2PC agent. It emphasizes the capabilities of multi-head Ptr-Net in situations where a large number of dimensions and combinations are involved. In summary, one of the main advantages of this methodology is its capability to be implemented at different sizes and approximate nearly optimal solutions while requiring reasonable computational resources.

TABLE 8. **Average performance improvement from A2PC in relation to GMwES.**

Scenario	A2PC outperformance*
SS	3.93%
SD	3.79%
LS	4.81%
LD	4.32%

*Measured in percentage.

V. CONCLUSION AND FUTURE WORK

This study presented a novel framework for addressing the GMEVTP problem. The core of this solution relies on a DRL agent based with multi-head Ptr-Net. The A2PC agent showed outstanding performance, even when compared with optimal values provided by the upper bound. In addition to validate the framework, these results demonstrate the power of neural-network branching, particularly in cases in which the dimensions and combinations become excessively large. Besides that, the agent architecture promotes the scaling capabilities of the learned models, which was confirmed using the same model with different sizes in the same topology scenarios.

After all, when compared with other methods, such as MIPOPT and GMwES, A2PC has emerged as a viable option for network design and planning. The proposal's strengths are its reasonable computation requirements and ability to reuse the same trained agent for different scales of the same problem.

Potential extensions of this work should focus on the organic expansion of utilizing this strategy in scenarios that demand multi-agent interactions [59] and continuous action domains. For example, investigating concerns related to energy savings. In that case, further enhancements may be achieved by investigating the agent's ability to eliminate or leverage cluster overlaps. Furthermore, this approach can be applied to address other complex problems that require the integration of multiple action dimensions. Notably, this is not limited to the specific context of LoRaWAN technologies. Instead, it provides an opportunity to utilize this framework with various technologies and for a wide range of problems.

REFERENCES

[1] B. Mendes, N. Correia, and D. Passos, "On the optimization of LoRaWAN gateway placement in wide area monitoring systems," in *Proc. IFIP Adv. Inf. Commun. Technol.*, 2022, pp. 41–51.

[2] V. Agarwal, S. Tapaswi, and P. Chanak, "A survey on path planning techniques for mobile sink in IoT-enabled wireless sensor networks," *Wireless Pers. Commun.*, vol. 119, no. 1, pp. 211–238, Jul. 2021.

[3] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "A survey on LoRa for IoT: Integrating edge computing," in *Proc. Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2019, pp. 295–300.

[4] A. Pagano, D. Croce, I. Tinnirello, and G. Vitale, "A survey on LoRa for smart agriculture: Current trends and future perspectives," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3664–3679, Feb. 2023.

[5] M. G. Ikhsan, M. Y. A. Saputro, D. A. Arji, R. Harwahyu, and R. F. Sari, "Mobile LoRa gateway for smart livestock monitoring system," in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IOTAIS)*, Nov. 2018, pp. 46–51.

[6] J. Fang, Z. Zhou, S. Jin, L. Wang, B. Lu, and Z. Qin, "Exploring LoRa for drone detection," in *Proc. IEE Conf. Comput. Commun. Workshops*, 2022, pp. 1–2.

[7] D. Mugerwa, N. Youngju, C. Hyunseok, S. Yongje, and L. Euisin, "SF-partition-based clustering and relaying scheme for resolving near–far unfairness in IoT multihop LoRa networks," *Sensors*, vol. 22, no. 23, p. 9332, 2022.

[8] R. Carvalho, N. Correia, and F. Al-Tam, "Q-learning ADR agent for LoRaWAN optimization," in *Proc. Artif. Intell., Commun. Technol. (IEEE IAICT)*, 2021, pp. 1041–1048.

[9] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell. 13th Innov. Appl.*, 2018, pp. 1–8.

[10] P. Kalmbach, J. Zerwas, P. Babarczi, A. Blenk, W. Kellerer, and S. Schmid, "Empowering self-driving networks," in *Proc. ACM SIGCOMM Afternoon Workshop Self-Driving Netw. (SelfDN)*, 2018, pp. 8–14.

[11] L. Csurgai-Horváth and J. Z. Bitó, "Route planning for mobile IoT devices," in *Proc. 11th Int. Symp. Commun. Syst. (CSNDSP)*, 2018, pp. 1–6.

[12] R. Carvalho, N. Correia, and F. Al-Tam, "Mobility planning of LoRa gateways for edge storage of IoT data," *Comput. Netw.*, vol. 221, Feb. 2023, Art. no. 109521.

[13] K. Almi'ani, A. Viglas, and L. Libman, "Mobile element path planning for time-constrained data gathering in wireless sensor networks," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 843–850.

[14] S. Gutiérrez et al., "Smart mobile LoRa agriculture system based on Internet of Things," in *Proc. IEEE 39th Central Amer. Panama Conv. (CONCAPAN)*, 2019, pp. 1–6.

[15] Z. Zhao et al., "A joint communication and computation design for distributed RIS-assisted probabilistic semantic communication in IIoT," *IEEE Internet Things J.*, vol. 11, no. 16, pp. 26568–26579, Aug. 2024.

[16] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for UAV-enabled mobile relaying systems," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4983–4996, Dec. 2016.

[17] S. Zhang, H. Zhang, Q. He, K. Bian, and L. Song, "Joint trajectory and power optimization for UAV relay networks," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 161–164, Jan. 2018.

[18] Y. Chen, W. Feng, and G. Zheng, "Optimum placement of UAV as relays," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 248–251, Feb. 2018.

[19] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-D placement of an unmanned aerial vehicle base station (UAV-BS) for energy-efficient maximal coverage," *IEEE Wireless Commun. Lett.*, vol. 6, no. 4, pp. 434–437, Aug. 2017.

[20] E. Montero et al., "Proactive radio- and QoS-aware UAV as BS deployment to improve cellular operations," *Comput. Netw.*, vol. 200, Dec. 2021, Art. no. 108486.

[21] H. Huang and A. V. Savkin, "Deployment of heterogeneous UAV base stations for optimal quality of coverage," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16429–16437, Sep. 2022.

[22] S. A. Al-Ahmed, M. Z. Shakir, and S. A. R. Zaidi, "Optimal 3D UAV base station placement by considering autonomous coverage hole detection, wireless backhaul and user demand," *J. Commun. Netw.*, vol. 22, no. 6, pp. 467–475, Dec. 2020.

[23] V. Saxena, J. Jaldén, and H. Klessig, "Optimal UAV base station trajectories using flow-level models for reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 4, pp. 1101–1112, Dec. 2019.

[24] H. Lu, X. Wei, H. Qian, and M. Chen, "A cost-efficient elastic UAV relay network construction method with guaranteed QoS," *Ad Hoc Netw.*, vol. 107, Oct. 2020, Art. no. 102219.

[25] İ. Baştürk, "Energy-efficient communication for UAV-enabled mobile relay networks," *Comput. Netw.*, vol. 213, Aug. 2022, Art. no. 109071.

[26] M. Behjati, A. B. M. Noh, H. A. H. Alobaidy, M. A. Zulkifley, R. Nordin, and N. F. Abdullah, "LoRa communications as an enabler for Internet of Drones towards large-scale livestock monitoring in rural farms," *Sensors*, vol. 21, no. 15, p. 5044, Jul. 2021.

[27] L. Sciullo, A. Trotta, and M. D. Felice, "Design and performance evaluation of a LoRa-based mobile emergency management system (LOCATE)," *Ad Hoc Netw.*, vol. 96, Jan. 2020, Art. no. 101993.

[28] G. K. Ijemaru, L. M. Ang, and K. P. Seng, "Transformation from IoT to IoV for waste management in smart cities," *J. Netw. Comput. Appl.*, vol. 204, Aug. 2022, Art. no. 103393.

[29] M. Stellin, S. Sabino, and A. Grilo, "LoRaWAN networking in mobile scenarios using a WiFi mesh of UAV gateways," *Electronics*, vol. 9, no. 4, p. 630, Apr. 2020.

[30] O. A. Amodu, U. A. Bukar, R. A. Raja Mahmood, C. Jarray, and M. Othman, "Age of information minimization in UAV-aided data collection for WSN and IoT applications: A systematic review," *J. Netw. Comput. Appl.*, vol. 216, Jul. 2023, Art. no. 103652.

[31] J. Dias and A. Grilo, "LoRa WAN multi-hop uplink extension," in *Proc. Int. Conf. Ambient Syst. (ANT)*, 2018, pp. 1–12.

[32] K. Nakamura et al., "A LoRa-based protocol for connecting IoT edge computing nodes to provide small-data-based services," *Digit. Commun. Netw.*, vol. 8, no. 3, pp. 257–266, Jun. 2022.

[33] Z. Zhao, Z. Yang, M. Chen, Z. Zhang, and H. V. Poor, "A joint communication and computation design for probabilistic semantic communications," *Entropy*, vol. 26, no. 5, p. 394, Apr. 2024.

[34] W. Xu, Z. Yang, D. W. K. Ng, M. Levorato, Y. C. Eldar, and M. Debbah, "Edge learning for B5G networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 9–39, Jan. 2023.

[35] I. Khoufi, A. Laouiti, C. Adjih, and M. Hadded, "UAVs trajectory optimization for data pick up and delivery with time window," *Drones*, vol. 5, no. 2, p. 27, Apr. 2021.

[36] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1–10.

[37] H. Shuai, F. Li, H. Pulgar-Painemal, and Y. Xue, "Branching dueling Q-network-based online scheduling of a microgrid with distributed energy storage systems," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 5479–5482, Nov. 2021.

[38] F. Wei et al., "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2197–2211, Dec. 2020.

[39] F. Al-Tam et al., "Deep PC-MAC: A deep reinforcement learning pointer-critic media access protocol," in *Proc. IEEE 25th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw. (CAMAD)*, 2020, pp. 1–6.

[40] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–9.

[41] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to schedule (LEASCH): A deep reinforcement learning approach for radio resource scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108088–108101, 2020.

[42] N. Correia et al., "Radio resource scheduling with deep pointer networks and reinforcement learning," in *Proc. IEEE Int. Workshop Comput. Aided Modeling Design (CAMAD)*, 2020, pp. 1–6.

[43] M. Volodymyr et al., "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[44] R. M. Sandoval, A.-J. Garcia-Sanchez, and J. Garcia-Haro, "Optimizing and updating LoRa communication parameters: A machine learning approach," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 884–895, Sep. 2019.

[45] R. M. Sandoval, A.-J. Garcia-Sanchez, J. Garcia-Haro, and T. M. Chen, "Optimal policy derivation for transmission duty-cycle constrained LPWAN," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3114–3125, Aug. 2018.

[46] N. A. El-Sherbeny, "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods," *J. King Saud Univ. Sci.*, vol. 22, no. 3, pp. 123–131, Jul. 2010.

[47] LoRa Alliance Team. (2017). *Lorawan Specification V1.1*. Accessed: Jun. 6, 2024. [Online]. Available: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1

[48] B. Irwan and H. Pham, "Neural combinatorial optimization with reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–5.

[49] A. Mazayev, F. Al-Tam, and N. Correia, "Attention-based model and deep reinforcement learning for distribution of event processing tasks," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100563.

[50] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–12.

[51] E. Tsironi, P. Barros, C. Weber, and S. Wermter, "An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition," *Neurocomputing*, vol. 268, pp. 76–86, Dec. 2017.

[52] R. Z. Carvalho. (2023). *LoRaRL-NS3 Agents*. Accessed: Jun. 6, 2024. [Online]. Available: https://github.com/rzuolo/NS3-LoraRL

[53] D. Magrin. (2019). *LoRa NS3 Module*. Accessed: Jun. 6, 2024. [Online]. Available: https://github.com/signetlabdei/lorawan

[54] J. M. Marais et al., "A review of LoRaWAN simulators: Design requirements and limitations," in *Proc. Int. Multidisciplinary Inf. Technol. Eng. Conf.*, 2019, pp. 1–6.

[55] Unmanned Aircraft Systems. *Top 10 Drones With the Longest Flight Time in 2024*. Accessed: Jun. 6, 2024. [Online]. Available: https://www.jouav.com/blog/drone-with-longest-flight-time.html

[56] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and unknown facts of LoRa: Experiences from a large-scale measurement study," *ACM Trans. Sensor Netw.*, vol. 15, no. 2, pp. 1–35, May 2019.

[57] T. Attia, M. Heusse, B. Tourancheau, and A. Duda, "Experimental characterization of LoRaWAN link quality," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[58] A. Rahmadhani and F. Kuipers, "When LoRaWAN frames collide," in *Proc. 12th Int. Workshop Wireless Netw. Testbeds, Experim. Eval. Characterization, Co-Located MobiCom*, 2018, pp. 89–97.

[59] W. Suttle, Z. Yang, K. Zhang, Z. Wang, T. Başar, and J. Liu, "A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1549–1554, 2020.