Federated Learning with a Drone Orchestrator

Path Planning for Minimized Staleness

Donevski, Igor; Babu, Nithin; Nielsen, Jimmy Jessen; Popovski, Petar; Saad, Walid

Published in:
IEEE Open Journal of the Communications Society

DOI (link to publication from Publisher):
10.1109/OJCOMS.2021.3072003

Creative Commons License
CC BY 4.0

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Received 15 March 2021; accepted 30 March 2021. Date of publication 12 April 2021; date of current version 3 May 2021.

Digital Object Identifier 10.1109/OJCOMS.2021.3072003

Federated Learning With a Drone Orchestrator: Path Planning for Minimized Staleness

IGOR DONEVSKI[1] (Graduate Student Member, IEEE), NITHIN BABU[1,2] (Graduate Student Member, IEEE), JIMMY JESSEN NIELSEN[1] (Senior Member, IEEE), PETAR POPOVSKI[1] (Fellow, IEEE), AND WALID SAAD[3] (Fellow, IEEE)

[1]Department of Electronic Systems, Aalborg University, 9220 Aalborg, Denmark

[2]SWIFT Lab, Research, Technology and Innovation Network, ALBA, The American College of Greece, 153 42 Athens, Greece

[3]Wireless@VT, Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

CORRESPONDING AUTHOR: I. DONEVSKI (e-mail: igordonevski@es.aau.dk)

This work was supported in part by European Union's Research and Innovation Programme through the Marie Sklodowska-Curie Grant under Agreement 812991 "PAINLESS" within the Horizon 2020 Program and in part by U.S. National Science Foundation under Grant CNS-1836802.

ABSTRACT In this paper, we investigate the problem of scheduling transmissions for spatially scattered nodes that contribute to a collaborative federated learning (FL) algorithm via wireless links provided by a drone. In the considered system, the drone acts as an orchestrator, coordinating the transmissions and the learning schedule within a predefined deadline. The actual schedule is reflected in a planned path: as the drone traverses it, it controls the distance and thereby the data rate to each node. Hence, the model is structured such that the drone orchestrator uses the path (trajectory) as its only tool to achieve fairness in terms of learning *staleness*, which reflects the learning time discrepancy among the nodes. Using the number of learning epochs performed at each learner as a performance indicator, we combine the average number of epochs computed and staleness into a balanced optimization criterion that is agnostic to the underlying FL implementation. We consider two methods for solving the complex trajectory planning optimization problem for static nodes: (1) successive convex programming (SCP) and (2) deep reinforcement learning (RL). Considering the proposed criterion, both methods are compared in three specific scenarios with few nodes. The results show that drone-orchestrated FL outperforms an immobile deployment by providing improvements in the range of 57% to 87.7%. Additionally, RL-guided trajectories are generally superior to SCP provided ones for complex node arrangements.

INDEX TERMS Drone trajectory optimization, wireless communications, federated learning, drone small cells, staleness minimization, reinforcement learning, convex approximation, unmanned aerial vehicles, edge computing.

I. INTRODUCTION

INITIALLY meant for military uses, then followed by a boom in commercial entertainment usage, the flying drones or unmanned aerial vehicles (UAVs) have a growing importance in the world of communications. The use of drones for wireless communication purposes has also received a surge of attention [1], [2] due to their excellent coverage to outdoor users. In particular, due to their flexibility, low altitude platform (LAP) drones are useful to act as on-demand drone small cells (DSCs) for wireless communication support. DSCs have the potential to continuously relocate while providing service to spatially scattered nodes that are unable to establish a high bandwidth ground-to-ground communication. Adhering to the surge of interest in enabling connected intelligence [3], [4], one can envision the use of such aerial platforms as an effective means to implement *collaborative federated learning (FL)*, where few geographically scattered nodes collaborate to improve a common machine learning (ML) model. Such an FL use case requires periodic and high bandwidth communications

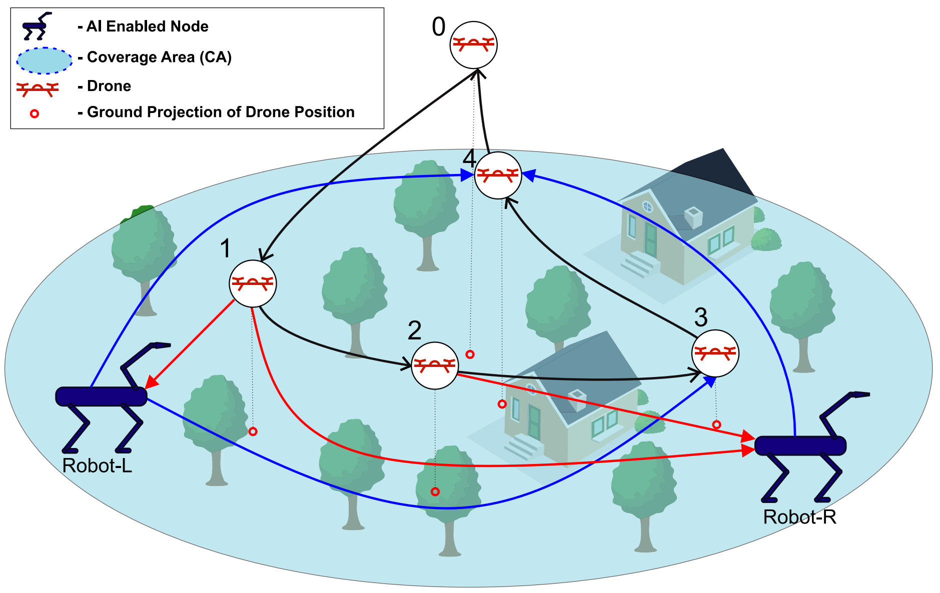

FIGURE 1. Toy illustration showcasing ML mode passing to the learners (red arrows) in the downloading (DL) phase, that then return the updated models in the uploading (UL) phase to achieve FL schedule as in Fig. 2. The drone performs this only by adjusting its trajectory, that in this example starts and finishes at the center of the CA.

to take place according to a given schedule. The planning of this schedule is directly dependent on the DSC's position, and it imposes various challenges. Moreover, assuming that all learners in an FL contain non-IID data that is useful to the overall model causes asynchrony between the learning input (epochs computed) of each node. Therefore, we consider the drone in the role of an *orchestrator*, whose trajectory is the available degree of freedom that can be adjusted to minimize the maximum *staleness* (difference between most ML epochs and the least ML epochs computed at a learner in the FL).

A. STATE OF THE ART

The effects of dynamic DSCs that move in favor of users' locations has been investigated in [5]–[7], where the focus is on the superior spectral efficiency and latency achieved by DSCs in various scenarios. In our previous work [8], we demonstrated the impact of accounting for the dynamic movements of standalone DSC, equipped with a tilting directional antenna. Moreover, the work in [9] focused on the energy efficiency for DSC deployment, while the authors in [10] and [11] studied the problem of placement optimization of a single cell and interference-limited multi DSC deployments, respectively.

However, the consideration of dynamic drones calls for efficient trajectory planning. The work in [12] focused on finding the drone trajectory that achieves minimal UAV energy expenditure while serving multiple nodes with a rotary-wing UAV. In [13], the authors investigated energy efficiency by scheduling the sleep timers of ground wireless nodes as well as the UAV's trajectory. Purpose-first three-dimensional trajectory design can be achieved as done in [14], whose goal was to maximize the minimum average data collection rate from all nodes for a stochastic channel model. Finally, considering the complexity of the issue, there is a strong incentive of solving trajectory optimization problems with the use of reinforcement learning (RL), as done in [15]–[17].

Drone cloudlet implementations receive a growing attention for edge computation purposes and offering portable processing services. In this setting, the works in [18] and [19] investigated the design of a drone trajectory, along with the problem of communications and computational resource allocation in favor of lowering the energy consumption of an Internet of Things (IoT) network. Combining this with the recent works on efficient offloading of the learning for RL, [20], sparked a new demand for drone-aided intensive edge computations. In a common centralized ML implementation, the drone would act as a sink for all the collected data which is then processed, as in the cloudlet design [18]. Additionally, FL implementations in drones have been a topic of significant interest in the literature. These prior works mostly consider the drones as learners [21]–[24]. However, even with current advances in energy efficient FL [25] and low power computation systems,[1] we consider the concept of drone-mounted ML-computation hardware as heavy and energy inefficient, thus reducing the flight time of the UAV.

B. DRONE ORCHESTRATOR FOR REDUCING STALENESS

The progress of robot implementations for laborious tasks in remote locations motivates investigating a setting of connected intelligence. In example, logging robots [26] that proceed with their main task of woodcutting can enhance their detection performance of critical flora and fauna through a collaborative learning process. In Fig. 1, we show two distant ground robot-nodes L and R that are not energy

1. https://www.dji.com/dk/manifold-2

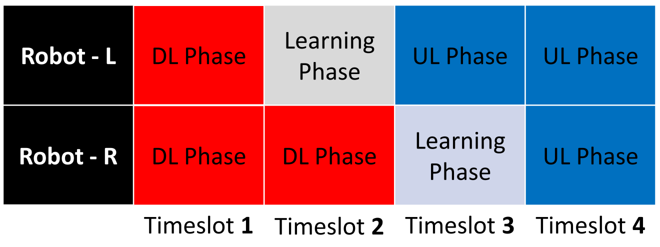

FIGURE 2. FL learners perform one iteration of learning thanks to balanced DL and UL phases.

restricted and are equipped with powerful ML computation equipment, and due to their spatial arrangement, require the communications support of a mobile drone. In this setting, both the learning and the sensing are distributed to the robot nodes that perform iterative improvements on a common ML model [27] with non-IID data [28] collected from their own sensors. As such, the drone assumes the role of an FL orchestrator that receives the model updates and aggregates them. This is a computational task that does not require powerful processors [27]. Such a setup sees potential practical implementations such as orchestrating automated agriculture, forestry, personalized healthcare [26], and border control [29] operations. Finally, this architecture can apply to a plethora of ML implementations, such as multi-perspective computer vision, multi-agent utility optimization, or semi-supervised parameter estimation. If needed, these implementations can also exploit the ability of FL to conceal sensitive information collected by the nodes.

In Fig. 1 and Fig. 2 we illustrate how each robot-node goes trough three phases: downloading (DL) model (red), learning, and uploading (UL) model improvements (blue). Transmission times to (DL) and from (UL) each node occupy useful time periods that would be preferably allocated for the learning phase at each robot. In addition, the learning at each node is impacted by its processing capability; higher processing capability at a node, with regards to other learners in the network, makes the data collected by its sensors more dominant when constructing the common model. Hence, the objective is to control the channel to each node through the drone-orchestrator's trajectory with the goal to aid the slow learning nodes (low processing capability) by modifying the DL and UL transmission times in order to minimize the work/learning discrepancies between nodes, called *staleness*.

Staleness is a cardinal metric for our setup since all nodes are assumed to possess useful data and, therefore, stragglers cannot be dropped. This creates asynchrony between the amount of learning each robot does. We model this asynchrony by the largest difference of epochs computed among the learners, which has been shown to be key for the performance of the next generation of asynchronous FL [30]–[32] The maximum staleness comes as a consequence of the asynchrony of such an FL implementation which is an issue that we want to tackle by implementing path planning in the duration of a single FL round. Solving the issue of staleness by only controlling the drone's trajectory requires new approaches as the transmissions occur with variable duration and only at the head and the tail of a pre-planned trajectory. This is in contrast to most works that are concerned with trajectory optimization problems whose goal is to maximize/minimize metrics that are often a direct representation of the aggregate rate instead of the fairness.

C. MAIN CONTRIBUTIONS AND ORGANIZATION

The main contribution of this paper is to develop, to the best of our knowledge, the first framework that designs a drone trajectory for serving FL networks with the purpose of customizing per node metrics across longer time periods, thus addressing the problem of *staleness* for deployments of scattered nodes. Addressing staleness gives a unique optimization criterion for the trajectory problem since it tackles shortening the transmission periods while equalizing work across many learners. Such a challenge makes our problem significantly different from prior works on trajectory optimization because of the combination of non-linear resource (channel quality across a trajectory) for solving a combinatorial optimization problem. It is apparent that minimizing staleness is not only limited for FL uses, but it also applies to drone trajectory design for lowering working-time discrepancies for IoT fields, data freshness for drones data-sink, and various edge computation scenarios where imbalanced work may harm the outlook of the implementation and the efficiency of the underlying service.

Section II describes the considered system, introduces the wireless communications traffic model of the FL implementation, and proposes a novel metric that contributes towards a balance between the total amount of epochs computed and *staleness*. We further compare the performance of two trajectory optimization approaches, successive convex programming and reinforcement learning, both detailed in Sections III and IV, respectively. The SCP requires that we approximate several metrics in order to get to a solvable form in a computationally efficient manner. The RL approach [33] makes no such demands and has the potential for future use in a stochastic and unpredictable environment. Moreover, a key novelty here is the fact that we created a more distance efficient hexagonal trajectory map and introduced a secondary RL experience buffer with only good memories that balances convergence and exploration. In Section V we compare and analyze both approaches. Finally, conclusions are drawn in Section VI.

II. SYSTEM MODEL

We consider a geographical coverage area (CA) defined by a circular range with radius $D_{\max}$, containing a set of K spatially distributed static nodes with each scattered node labeled by $k \in \mathcal{K} = \{1, 2, \ldots, K\}$. A single drone-orchestrator travels across this CA and has to complete its route within a predefined deadline T referred to as global cycle clock. The global cycle clock is defined by the FL implementation and it is crucial to the correct operation of an underlying task. Note that in both FL algorithms FedAvg [27] and FedProx [28] it is critical that not too much local work (very long T) takes place as it impairs the aggregate learning. On the other hand, too little local work (short T) means the transmission overhead will become dominant [27]. We further define N discrete timeslot intervals $i \in \mathcal{N} = \{1, 2, \ldots, N\}$ each with a duration of Δ seconds as $T = N\Delta$. The size of Δ is determined based on the drone speed and CA radius as discussed in Section IV and Section V. The drone trajectory $\mathbf{P}^{\mathrm{d}} = \{\mathbf{p}_i^{\mathrm{d}}\} \in \mathbb{R}^{N\times 2}$, where each $\mathbf{p}_i^{\mathrm{d}}$ represents a way-point for each timeslot i as the i-th row of $\mathbf{P}^{\mathrm{d}}$ is decribed by the horizontal coordinates $\mathbf{p}_i^{\mathrm{d}} = (x_i^{\mathrm{d}}, y_i^{\mathrm{d}})$, while its altitude is always H. The drone can fly horizontally, limited by a maximum speed of $v_{\max}$, and it is equipped with a directional antenna that tilts to ensure coverage over the CA as in [34], so that basic control signaling is always supported. This does not imply satisfactory FL model transfer rate for distant users, making it necessary to move the drone closer to ensure faster model exchange. The K nodes are found on the ground (zero height) at positions $\mathbf{p}^k = (x^k, y^k)\ \forall k \in \mathcal{K}$, resulting in a drone-to-node k horizontal distance of:

$$d_{i,k}\left(\mathbf{p}_i^{\mathrm{d}}\right) = \sqrt{\left(x_i^{\mathrm{d}} - x^k\right)^2 + \left(y_i^{\mathrm{d}} - y^k\right)^2} \tag{1}$$

where $k \in \mathcal{K}$. As such, the value of $d_{i,k}$ is subject to the anticipated movement of the drone at time i, which further impacts the data rate $R_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ for each node. We do not adjust drone's height during the trajectory, as it is incompatible with the use of a directional antenna in a sense that it would affect the size of the CA (e.g., see [1], [2], and [8]). Moreover, adjusting the height can raise potential liability concerns with respect to collisions, as well as drastically increase the optimization complexity. We also consider localization precision defined by a radius of r meters around the allocated drone position $\mathbf{p}_i^{\mathrm{d}}$.

A. FEDERATED LEARNING TRAFFIC MODEL

At the start of the FL cycle, each node starts with the DL phase that lasts $N_{k,\mathrm{DL}}(\mathbf{P}^{\mathrm{d}})$ timeslots, implicitly given as:

$$\sum_{i=0}^{N_{k,\mathrm{DL}}(\mathbf{P}^{\mathrm{d}})} R_{i,k}\left(\mathbf{p}_i^{\mathrm{d}}\right) \cdot \Delta = B \quad \forall k \in \mathcal{K} \tag{2}$$

where $R_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ is the instantaneous data rate for timeslot i of node k. Each node concludes the FL cycle with the UL phase of duration $N_{k,\mathrm{UL}}(\mathbf{P}^{\mathrm{d}})$ implicitly given by:

$$\sum_{i=N-N_{k,\mathrm{UL}}(\mathbf{P}^{\mathrm{d}})}^{N} R_{i,k}\left(\mathbf{p}_i^{\mathrm{d}}\right) \cdot \Delta = B \quad \forall k \in \mathcal{K}. \tag{3}$$

The leftover time in between both transmission phases is where the learning occurs for each learner. Since all phases need to be completed within N timeslots, the discrete learning period $N_{k,\mathrm{L}}(\mathbf{P}^{\mathrm{d}})$ during the drone's trajectory lasts:

$$N_{k,\mathrm{L}}\left(\mathbf{P}^{\mathrm{d}}\right) = N - N_{k,\mathrm{DL}}\left(\mathbf{P}^{\mathrm{d}}\right) - N_{k,\mathrm{UL}}\left(\mathbf{P}^{\mathrm{d}}\right) \quad \forall k \in \mathcal{K}. \tag{4}$$

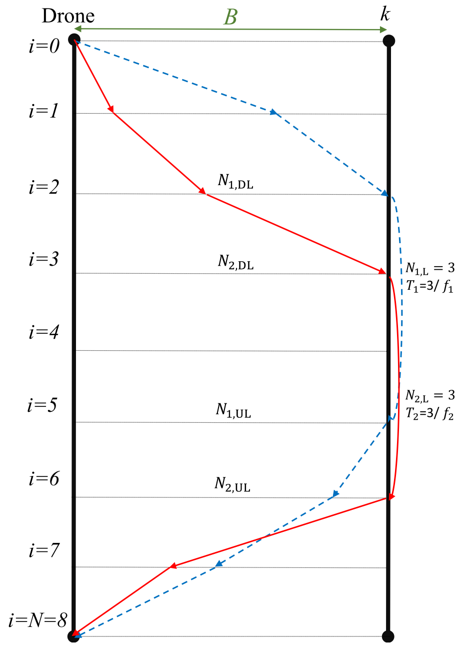

FIGURE 3. An example of the FL process with two learners; $k = 1$ blue-dashed line and $k = 2$ red-solid line. Here $N_{1,\mathrm{L}} = N_{2,\mathrm{L}} = 3$, and if $f_1 = f_2$, the system has zero ***staleness*** since both nodes have an equal amount of epochs.

Fig. 3 illustrates the phases for $K = 2$ where the ML process is based on a model of size B Mb which, for simplicity, is assumed to be identical for both UL and DL. The illustration portrays a two-dimensional system that has a width of B Mbs and height of the number of discrete timeslots N. Each one of the dashed-blue ($k = 1$) and solid-red ($k = 2$) lines represent the FL stage. Meanwhile, the slope of the lines capture $R_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ that changes across all timeslots because the drone moves horizontally and impacts the wireless channel.

However, ML is an iterative process where each learning pass of the sensed data is called an epoch T_k. Back in Fig. 3, the allocated processing time for each node is three slots, and even though we mitigate the asynchrony introduced by the spatial arrangement, the computed epochs T_k depend on the processing capabilities of node k. To account for such learners, referred to as *stragglers* in the FL community, we scale the computational capability with a scalar value f_k. The value f_k represents the amount of epochs processed per second and is a collective measure of the processor cores and speed, ML accelerator (such as repurposed rasterisation cores) and/or the environment sampling rate. The number of epochs spent by device k is therefore a function of the flown trajectory $\mathbf{P}^{\mathrm{d}}$ and is calculated as:

$$T_k\left(\mathbf{P}^{\mathrm{d}}\right) = N_{k,\mathrm{L}}\left(\mathbf{P}^{\mathrm{d}}\right) \cdot \Delta \cdot f_k. \tag{5}$$

In classical ML, the more epochs computed and the more data is provided, the higher the expected model accuracy. FL is a cyclic process whereby at the end of the cycle there are K models with different weights $\mathrm{w_k}$ that are received by the drone orchestrator and are aggregated to the common model [27], [28]. The orchestrator then initiates a new learning cycle and returns the aggregated model weights $\mathrm{w_d}$ to all participating learners/nodes.

B. STALENESS IN A NO-DROP FEDERATED LEARNING

Since in an FL the data is non-IID distributed among learners, we model the performance of the FL as a no-drop FL where learning occurs in an asynchronous fashion and all participants are trustworthy. Like this, discrepancies in T_k between the different learners, i.e., $|T_k(\mathbf{P}^{\mathrm{d}}) - T_l(\mathbf{P}^{\mathrm{d}})|;\ \forall k \neq l;\ k, l \in \mathcal{K}$, are referred to as *staleness* [31]. This creates an asynchronous FL where staleness undermines the total learning done $\sum_{k=1}^{K} T_k$ and therefore slows down FL convergence and lowers overall system accuracy in the training phase [30]–[32]. Moreover, both [28] that tackles learning without dropping straggles, and the asynchronous optimization [31] work, show that even when the local optimizer is designed for some asynchronous amount of work the maximum staleness impacts the performance of the FL. Hence, we aim to improve learning performance when aggregating the collective model by minimizing the largest epoch number difference between any two learners:

$$s\left(\mathbf{P}^{\mathrm{d}}\right) = \max\left(\left|T_k\left(\mathbf{P}^{\mathrm{d}}\right) - T_l\left(\mathbf{P}^{\mathrm{d}}\right)\right|\right); \quad \forall k \neq l;\ k, l \in \mathcal{K}, \tag{6}$$

where T_k is relaxed to $T_k \in \mathbb{R}^+$, for the purpose of generalizing the analysis. To avoid fully neglecting good learners, we introduce the mean of the total number of epochs performed as a stabilizing factor. We can now define the primary optimization criterion for our drone trajectory as an average-anchored staleness (AAS):

$$\max_{\mathbf{P}^{\mathrm{d}}} \ \frac{1}{K}\sum_{k=1}^{K} T_k\left(\mathbf{P}^{\mathrm{d}}\right) - s\left(\mathbf{P}^{\mathrm{d}}\right), \tag{7}$$

where the system constraints of speed, initial location, and deadline are described in Section III for the SCP approach and Section IV for the RL approac. As mentioned, different local FL optimizers can tolerate some asynchronous amount of work. Such tolerance s_{tol} can be accounted for in (7) by converting the second term $s(\mathbf{P}^{\mathrm{d}})$ to $\max(s(\mathbf{P}^{\mathrm{d}}), s_{\mathrm{tol}})$. Although the implementation in [32] would moderately tolerate maximum staleness of 4, we proceed the work by tackling the most challenging scenario where the trajectory would need to be optimized if no tolerance was allowed $s_{\mathrm{tol}} = 0$.

A more simplified look at (7) is that if node positioning is stochastic as in a point process, the goal of our optimization problem would map to reducing the maximum deviation in learning performed. This gives a good general overview that is data-agnostic [35], without the need to assume the impact of data at some particular learner and solely on spatial and computational performance. Therefore, AAS provides trajectories that serve an equally balanced amount of learning and *staleness*, which can be further enhanced with additional resource allocation techniques [36] combined with FL incentive calculations [37]. Finally, with (7) we bring asynchronous FL as close as reasonable by the disposable resources to a classical synchronous FL (where all learners compute the same amount of epochs in a cycle), without dropping any stragglers.

C. PROPAGATION ENVIRONMENT

A node can belong to one of two propagation groups, nodes that have direct line-of-sight (LoS) or no-LoS (NLoS). As such, the path loss experienced for node k at time i is $\ell_{i,k}$ and becomes a sum of the free space path loss (FSPL) and the additional large-scale shadowing coefficient for each one of the propagation groups. We note that since we are concerned with lengthy transmission timescales of several seconds we use the mean fading coefficients for each propagation group, namely η_{LoS} and η_{NLoS}. This provides a well-generalized approach opposed to working with random variables for the fading calculations of the normally distributed excessive path loss, that is introduced due to the large features of the topology [38]. Taking into account a directional antenna with directivity defined by G_t, the path loss experienced for each propagation group becomes:

$$\ell_{i,k,\mathrm{LoS}}\left(\mathbf{p}_i^{\mathrm{d}}\right) = -10\log(G_t) + 20\log\left(\sqrt{d_{i,k}^2\left(\mathbf{p}_i^{\mathrm{d}}\right) + H^2}\right) + C + \eta_{\mathrm{LoS}}, \tag{8}$$

and,

$$\ell_{i,k,\mathrm{NLoS}}\left(\mathbf{p}_i^{\mathrm{d}}\right) = -10\log(G_t) + 20\log\left(\sqrt{d_{i,k}^2\left(\mathbf{p}_i^{\mathrm{d}}\right) + H^2}\right) + C + \eta_{\mathrm{NLoS}}, \tag{9}$$

where log is a shortened version of the common logarithm $\log_{10}$ and the term C is a substitute for the carrier frequency f_c constant in FSPL $C = 20\log\left(\frac{f_c 4\pi}{c}\right)$. Hence, a node's probability to belong to either group is directly dependent on the probability for a LoS to happen, $\mathrm{P_{LoS}}(\mathbf{p}_i^{\mathrm{d}})$. To represent the probability we use the s-curve model defined by [39]:

$$P_{\mathrm{LoS}}\left(\mathbf{p}_i^{\mathrm{d}}\right) = \frac{1}{1 + a\exp\left(-b\left(\theta_{\mathrm{user}}(\mathbf{p}_i^{\mathrm{d}}) - a\right)\right)}, \tag{10}$$

where a and b are constants dependent on the topological setting, and the elevation angle at user side $\theta_{\mathrm{user}}(\mathbf{p}_i^{\mathrm{d}})$, expressed in degrees $0 \leq \theta_{\mathrm{user}}(\mathbf{p}_i^{\mathrm{d}}) \leq 90$, is a function of the drone's horizontal position since the height H is fixed and $\theta_{\mathrm{user}}(\mathbf{p}_i^{\mathrm{d}}) = \arctan\left(\frac{H}{d_{i,k}(\mathbf{p}_i^{\mathrm{d}})}\right)$, as illustrated in Fig. 4.

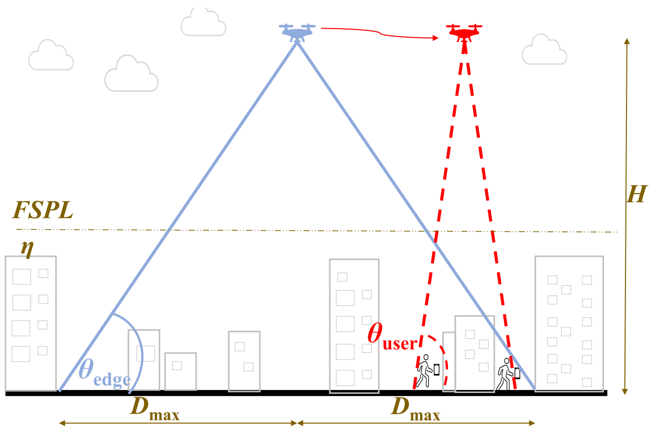

FIGURE 4. Reference LAP placement (solid blue) and a drone orchestrator in trajectory (dashed red) [8].

Given the existence of only two propagation groups, the probability $P_{\text{NLoS}}(\mathbf{p}_i^{\text{d}}) = 1 - P_{\text{LoS}}(\mathbf{p}_i^{\text{d}})$ which finally allows us to calculate the final path loss expectation $L_{i,k}(\mathbf{p}_i^{\text{d}})$ for node k as derived from (8), (9), and (10) and given in linear terms as a function of the drone's position through $\ell_{i,k,\text{LoS}}(\mathbf{p}_i^{\text{d}})$ and $\ell_{i,k,\text{NLoS}}(\mathbf{p}_i^{\text{d}})$:

$$10\log\Big[L_{i,k}\Big(\mathbf{p}_i^{\text{d}}\Big)\Big] = \ell_{i,k\text{LoS}}\Big(\mathbf{p}_i^{\text{d}}\Big)\cdot P_{\text{LoS}}\Big(\mathbf{p}_i^{\text{d}}\Big) \quad (11)$$

$$+\ \ell_{i,k\text{NLoS}}\Big(\mathbf{p}_i^{\text{d}}\Big)\cdot P_{\text{NLoS}}\Big(\mathbf{p}_i^{\text{d}}\Big) \quad (12)$$

$$= P_{\text{LoS}}\Big(\mathbf{p}_i^{\text{d}}\Big)(\eta_{\text{LoS}} - \eta_{\text{NLoS}}) + \ell_{\text{NLoS}}\Big(\mathbf{p}_i^{\text{d}}\Big). \quad (13)$$

D. DATA RATE

We consider a tilting antenna that keeps the whole CA in a communications coverage range and can provide a non-zero rate at any time. Such a setup reduces interference to users outside the CA [8] as well. The gain of the antenna G_t is given by its effectiveness E_r to fit an ideal conical beamwidth $G_t = E_r 10\log(G_I)$, where the ideal conical antenna has gain:

$$G_I = \frac{2}{1-\sin\left(\theta_{\text{edge}}\frac{\pi}{180}\right)}, \quad (14)$$

where $\theta_{\text{edge}} = \arctan(\frac{H}{D_{\max}})$ is the elevation angle at the cell edge, when the drone is positioned in the middle of the CA and applies no antenna tilt, as shown on Fig. 4. With this we define the final expected path loss expression, as a function of the location of the drone, and fully expanded [8]:

$$10\log\Big(L_{i,k}\Big(\mathbf{p}_i^{\text{d}}\Big)\Big) = \frac{\eta_{\text{LoS}} - \eta_{\text{NLoS}}}{1 + a\exp\left\{-b\left[\arctan\left(\frac{H}{d_{i,k}(\mathbf{p}_i^{\text{d}})}\right) - a\right]\right\}}$$
$$+\ 20\log\left(\sqrt{d_{i,k}^2(\mathbf{p}_i^{\text{d}}) + H^2}\right)$$
$$-\ E_r 10\log\left[\frac{2}{1-\sin\left(\theta_{\text{edge}}\frac{\pi}{180}\right)}\right]$$
$$+\ C + \eta_{\text{NLoS}}. \quad (15)$$

We consider an orthogonal multiple access scheme (e.g., using frequency division multiple access (FDMA)), thus, we consider a noise-limited system with no interference for both DL and UL phases. Moreover, to emphasize the importance of the drone position in its trajectory we preallocate a frequency spectrum W that remains constant for each node. Therefore, at each timeslot the achievable rate becomes:

$$R_{i,k}\Big(\mathbf{p}_i^{\text{d}}\Big) = W\log_2\left[1 + \frac{P_t}{WN_0 L_{i,k}(\mathbf{p}_i^{\text{d}})}\right], \quad (16)$$

where P_t is the transmission power that is assumed to be identical at both node and drone side, while N_0 is the noise spectral density linearly scaling the noise with the channel bandwidth W.

III. PROBLEM ANALYSIS AND CONVEX APPROXIMATION FOR TRAJECTORY OPTIMIZATION

In this section, we determine the optimal trajectory of the drone orchestrator in a way to minimize the discrepancy between the T_k values while maximizing each individual T_k value for a given mission completion time using the SCP technique. We use this method with the overarching goal of devising an algorithm that will provide a solution in deterministic polynomial time for any scenario with arbitrary arrangement of nodes. Due to the several non-convex parameters involved, this is non-trivial and requires a combination of several analytical techniques.

As defined in Section II, the value of the length of a timeslot $\Delta = \frac{T}{N}$ is selected such that any position reachable within that timeslot does not impose significant changes in the rate performance; i.e., can be considered to be approximately unchanged. Hence by first order Taylor approximation, at some time instant t, the discrete position of the drone at the time slot $t + \Delta$ can be approximated as:

$$\mathbf{p}_{i+1}^{\text{d}} = \mathbf{p}_i^{\text{d}} + \mathbf{v}_i^{\text{d}}\Delta + \frac{1}{2}\mathbf{a}_i^{\text{d}}\Delta^2 \quad \forall i \in \mathcal{N}. \quad (17)$$

Similarly, the velocity vector for the time slot $t + \Delta$ can be approximated as:

$$\mathbf{v}_{i+1}^{\text{d}} = \mathbf{v}_i^{\text{d}} + \mathbf{a}_i^{\text{d}}\Delta \quad \forall i \in \mathcal{N}. \quad (18)$$

The use of time discretization reduces the number of variables to $2\cdot N$, resulting in the following *staleness* minimization problem:

$$(\text{P1}): \underset{\mathbf{P}^{\text{d}}}{\text{maximize}} \quad \frac{1}{K}\sum_{k=1}^{K} T_k\Big(\mathbf{P}^{\text{d}}\Big) - s\Big(\mathbf{P}^{\text{d}}\Big), \quad (19)$$

$$\text{s.t.} \quad \Big|T_k\Big(\mathbf{P}^{\text{d}}\Big) - T_l\Big(\mathbf{P}^{\text{d}}\Big)\Big| \le s\Big(\mathbf{P}^{\text{d}}\Big), \quad k,l \in \mathcal{K}, \quad (20)$$

$$\frac{2BT}{\Delta\sum_{i=1}^{N} R_{i,k}(\mathbf{p}_i^{\text{d}})} + \frac{T_k(\mathbf{P}^{\text{d}})}{f_k} \le T \quad \forall k \in \mathcal{K}, \quad (21)$$

$$\Big\|\mathbf{p}_{i+1}^{\text{d}} - \mathbf{p}_i^{\text{d}}\Big\| \le \min\{2r, \Delta v_{\max}\}, \quad (22)$$

$$T_k\Big(\mathbf{P}^{\text{d}}\Big) \ge 1 \quad \forall k \in \mathcal{K}, \quad (23)$$

$$\mathbf{p}_1^{\text{d}} = \mathbf{p}_I, \quad (24)$$

$$\mathbf{v}_1^{\text{d}} = \mathbf{v}_I, \quad (25)$$

$$\mathbf{v}_N^{\text{d}} = \mathbf{v}_F, \quad (26)$$

$$\Big\|\mathbf{v}_i^{\text{d}}\Big\| \ge v_{\min} \quad \forall i, \quad (27)$$

$$(17),\ (18). \quad (28)$$

Maximizing the objective function (19) is equivalent to minimizing the maximum difference between the T_k values of the nodes in the CA. Equation (21) is the mission completion time constraint, where the first term accounts for the time required for transmitting both uplink and downlink the B bits of data. In the aforementioned term, we approximate the data rate during the UL and DL phases as the average rate through the full time T. This avoids having to convert the problem into a mixed-integer one. Constraint (22) represents the localization precision limitation by which the maximum distance between consecutive positions of the drone is limited to $2 \cdot r$ as given back in Section II; (24) is the initial drone position constraint; while (23) guarantees at least one iteration of the local ML model with the given data subset for every node.

It is evident that the average rate of (21) depends on $\mathbf{p}_i^{\mathrm{d}}$ through $L_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ and therefore the LoS probability. The complex expression of $P_{\mathrm{LoS}}(\mathbf{p}_i^{\mathrm{d}})$ as reported in (10) makes solving (P1) using convex methods outside our computational capabilities. To circumvent the impact of the Lo probability on function convexity, we use a homogeneous approximation for the LoS probability [12]. Indeed, since $P_{\mathrm{LoS}}(\mathbf{p}_i^{\mathrm{d}})$ is an increasing function of the elevation angle, we consider the LoS probability of all the nodes equal to the LoS probability of the edge user device [11], i.e., $P_{\mathrm{LoS}}(\mathbf{p}_i^{\mathrm{d}}) \approx P_{\mathrm{LoS}}(\mathbf{p}_{\mathrm{edge}}^{\mathrm{d}})\ \forall i, k : d_{i,k} \leq D_{\max}$. Hence the corresponding rate value is the lower bound of $R_{i,k}(\mathbf{p}_i^{\mathrm{d}})$; which in a LoS dominated region is equal to the actual $R_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ value. Therefore the lower bound of the average achievable rate of node k when the drone is at time i becomes:

$$\overline{R}_{i,k}\left(\mathbf{p}_i^{\mathrm{d}}\right) = W\log_2\left[1 + \frac{\gamma_o}{\left(H^2 + d_{i,k}^2\right)\overline{L}}\right], \tag{29}$$

where $\overline{L} = 10\{[P_{\mathrm{LoS}}(\mathbf{p}_{\mathrm{edge}}^{\mathrm{d}})(\eta_{\mathrm{LoS}} - \eta_{\mathrm{NLoS}}) + \eta_{\mathrm{NLoS}}]/10\}$ and $\gamma_o = \dfrac{P_t c^2 G_I^{E_r}}{WN_o(4\pi f)^2}$ are obtained by substituting (9) in (11). This rate-equivalent approximation is useful in providing solutions to the convex approximation approach but is not necessary in Section IV where we approach the same problem using reinforcement learning.

The objective function and the constraints (20), (22)-(26), (17), (18) are convex functions of the position and the velocity variable. However, the mission completion time constraint (21) is non-convex because of the data rate expression in (29) and the minimum velocity constraint is also a non-convex function of the velocity variable. Hence (19) cannot be solved directly by using a convex optimization technique. We will thus address this challenge by using the sequential convex programming technique.

A. SEQUENTIAL CONVEX PROGRAMMING FOR TRAJECTORY OPTIMIZATION

The SCP approach allows us to represent a non-convex optimization problem as a sequence of convex optimization problems, and, then, solve them iteratively until the solution converges [40]. The candidate solution obtained through the SCP is guaranteed to satisfy the Karush-Kuhn-Tucker (KKT) conditions of the actual non-convex problem. Hence the solution obtained through the SCP technique cannot be considered as the global optimum of the problem, but, instead, it is a local optimum. However, the rate of convergence of the SCP algorithm is linear in complexity thereby making it suitable for solving real-time drone positioning problems [40].

Algorithm 1: SCP for Trajectory Optimization

```
1 Input: $l = 0$; $\{\mathbf{p}_i^{\mathrm{d},l}\}$, $\{\mathbf{v}_i^{\mathrm{d},l}\}$ $\forall i \in \mathcal{N}$
2 repeat
3 Solve (P1.1) using available convex optimization tool
    box to obtain the optimal solution: $\{\mathbf{p}_{\mathrm{o},i}^{\mathrm{d}}\}$, $\{\mathbf{v}_{\mathrm{o},i}^{\mathrm{d}}\}$
4 $l = l + 1$;
5 $\mathbf{p}_i^{\mathrm{d},l} = \mathbf{p}_{\mathrm{o},i}^{\mathrm{d}}$; $\mathbf{v}_i^{\mathrm{d},l} = \mathbf{v}_{\mathrm{o},i}^{\mathrm{d}}$ $\forall i \in \mathcal{N}$
6 until The fractional increase in the objective function
    of (P1.1) is less than a threshold $\delta$
7 Output: Optimal drone positions: $\{\mathbf{p}_i^{\mathrm{d},l}\}$
```

To solve (19) using the SCP technique, we introduce an auxiliary variable $\lambda_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ to tackle the non-convex constraint (21); where $\lambda_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ is the first order Taylor approximation of $\overline{R}_{i,k}(\mathbf{p}_i^{\mathrm{d}})$ expressed as:

$$\lambda_{i,k}\left(\mathbf{p}_i^{\mathrm{d}}\right) \leq \log_2\left(1 + \frac{\gamma}{H^2 + \left\|\mathbf{p}_i^{\mathrm{d},l} - \mathbf{p}^k\right\|^2}\right) - \alpha_{k,i}^{\mathrm{d},l}\left(\left\|\mathbf{p}_i^{\mathrm{d}} - \mathbf{p}^k\right\|^2 - \left\|\mathbf{p}_i^{\mathrm{d},l} - \mathbf{p}^k\right\|^2\right), \tag{30}$$

where $\alpha_{k,i}^{\mathrm{d},l} = \dfrac{\gamma\log_2 e}{(H^2 + \gamma + \left\|\mathbf{p}_i^{\mathrm{d},l} - \mathbf{p}^k\right\|^2)(H^2 + \left\|\mathbf{p}_i^{\mathrm{d},l} - \mathbf{p}^k\right\|^2)}$; $\{\mathbf{p}_i^{\mathrm{d},l}\}$ is the set of drone positions obtained from the l^{th} iteration; $\gamma = \gamma_o/\overline{L}$. The right-hand side of (30) is a concave function of $\mathbf{p}_i^{\mathrm{d}}$; hence (30) is a convex constraint of the variables $\mathbf{p}_i^{\mathrm{d}}$ and $\lambda_{i,k}(\mathbf{p}_i^{\mathrm{d}})$. Similarly the non-convex minimum velocity constraint can be equivalently represented as;

$$\beta \geq v_{\min}, \tag{31}$$

$$\left\|\mathbf{v}_i^{\mathrm{d},l}\right\|^2 + 2\left(\mathbf{v}_i^{\mathrm{d}} - \mathbf{v}_i^{\mathrm{d},l}\right)\left(\mathbf{v}_i^{\mathrm{d},l}\right)^T \geq \beta^2, \tag{32}$$

with (30), (31), and (32), the optimization problem (19), can be equivalently written as;

$$\text{(P1.1)} : \underset{\mathbf{P}^{\mathrm{d}}}{\text{maximize}}\ \frac{1}{K}\sum_{k=1}^{K} T_k\left(\mathbf{P}^{\mathrm{d}}\right) - s\left(\mathbf{P}^{\mathrm{d}}\right), \tag{33}$$

$$\text{s.t.}\ \frac{2BT}{\Delta\sum_{i=1}^{N}\lambda_{i,k}\left(\mathbf{P}^{\mathrm{d}}\right)} + \frac{T_k\left(\mathbf{P}^{\mathrm{d}}\right)}{f_k} \leq T,\ \ k \in \mathcal{K}, \tag{34}$$

$$(17), (18), (20), (22)–(26), (31), (32). \tag{35}$$

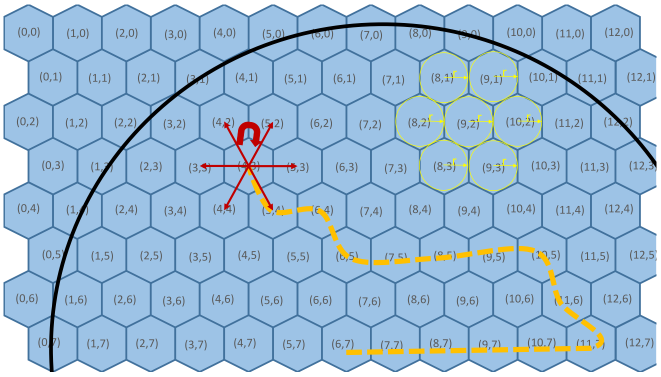

FIGURE 5. A cropped illustration of the *odd-r* hexagonal lattice (blue) fitting the CA's border (black solid line), with few examples of the precision circles with inner radius *r* (yellow). The drone has flown a trajectory (dashed orange line) and is faced with deciding among the 7 actions (solid red arrows).

Finally, the objective function and the constraints of (33) are convex and the problem can be iteratively solved as reported in Algorithm 1. In our case, every iteration makes use of available convex optimization tool box by MATLAB called CVX [41]. Finally, we note that the resulting trajectory may be a local maximum that is influenced by the initialization of the approximated trajectory positions. Nonetheless, this is sufficient given that SCP should provide reliably similar performances across different testing scenarios and executes in polynomial time.

IV. REINFORCEMENT LEARNING FOR TRAJECTORY OPTIMIZATION

In Section III, we analyzed (7) by converting it to a convex form and solving it iteratively as in SCP. Due to the shortcomings of SCP, we are interested in evaluating the performance and suitability of RL [33] particularly without making approximations on the original scenario proposed in Section II. In addition, RL has a potential to work with more difficult and realistic models that include stochastic rates, node positions, and variations in the drone speed, all of which can be suitable for future real-world variations of the problem. Given this motivation, we will now investigate how to design an RL that solves the trajectory problem as per the defined model in Section II. In general, RL is a learning approach that is used for finding the optimal way of executing a task by letting an entity, named agent, take actions that affect its state within the acting environment [33]. The agent improves over time by incorporating the rewards it had received for its appropriate performance in all episodes within that same environment.

A. ENVIRONMENT REMODELLING

We discretize the continuous environment by splitting the horizontal space into an *odd-r* horizontal layout hexagonal lattice, such as illustrated in Fig. 5. Beyond reducing the continuous space, the grid becomes a truthful representation of the trajectory limitations due to drone positioning. Setting r as the inner radius of each hexagon, shown with yellow at Fig. 5, does not harm the precision of the planned trajectory and offers superior packing. This also requires the position of each static node k defined by the center of the hexagon it is in. Using the nearest neighbouring center distance $2 \cdot r$ division of CA's diameter $2 \cdot D_{\max}$, results in a two dimensional $M \times M$ lattice of size:

$$M = \frac{D_{\max}}{r}. \tag{36}$$

We define as a single episode the completion of the deadline T during which all nodes need to have their updates sent back to the drone and is discretized by $\Delta = T/N$ for all N timeslots. The drone starts at point p_0^{d} and finishes at point p_N^{d} by travelling from point i to point $i+1$ with a maximum speed limitation of $\frac{p_{i+1}^{\mathrm{d}} - p_i^{\mathrm{d}}}{\Delta} \leq v_{\max}^{\mathrm{d}}$. Since the drone speed is limited, we allow for full trajectory resolution of the hexagonal lattice by setting:

$$\Delta = \frac{2 \cdot r}{v_{\max}^{\mathrm{d}}}, \tag{37}$$

resulting in a total number of timeslots as:

$$N = \frac{T \cdot v_{\max}^{\mathrm{d}}}{2 \cdot r}. \tag{38}$$

In this way, at every time-step, the drone has the choice to move to each of the six neighbouring hexagons or not move at all, illustrated with red at Fig. 5, totalling to an action space of 7. Since the trajectory is expressed by all previous drone movements the problem of trajectory planning becomes a Markov Decision Process (MDP) that under special conditions has a size of an N-tuple with base 7 totalling to 7^N states.

B. DRONE TRAJECTORY AS AN MDP

To make the trajectory problem solvable by RL, the main purpose of the reformulation until now was to make it representable by an MDP. To this end, we use a standard MDP representation as a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ with sets: state space $\mathcal{S}$, action space $\mathcal{A}$, probability of transition $\mathcal{P}$, and a state-action reward map $\mathcal{S} \times \mathcal{A} -> \mathcal{R}$.

- $\mathcal{S}$ - Each state in the set is defined by the drone's (x, y) coordinates in the drone trajectory taken, depicted with orange back in Fig. 5, the leftover timeslots that lack coordinates are padded totalling to $2 \cdot N$ states. We additionally specify each state with the number of leftover timeslots $N - i$, and (x^k, y^k), f_k, and $R_{i,k}$ for each node k. Thus we have a state described by $F = 1 + (2 \cdot N) + (4 \cdot K)$ features. Since the state space for the MDP scales with the number of timeslots N, the complexity of trajectory optimization becomes more challenging to the RL agent from two additional perspectives. A larger N directly increases the size of input features of the network F while it also extends the time for which the agent needs to finish a whole episode, providing less experience for the same time spent learning.

- $\mathcal{A}$ - The action space is defined by all possible movement directions on the sides of the hexagon plus the action of remaining in the same place formatted into a 7-tuple.
- $\mathcal{P}$ - Since the defined MDP is deterministic, no randomness is included in the set and all transitions follow the agent's decisions. Therefore, the next state is a direct consequence of the action that the agent takes, which is the maximal value among the seven outputs in the tuple.
- $\mathcal{R}$ - Since the performance of the drone can only be known once it completes the trajectory, defining continuous rewards for the agent will lead to a suboptimal performance due to the imbalance between the progressive rewards and the optimization criterion. Therefore, the reward set $\mathcal{R}$ is 0 except for the end of the episode where the one-time reward is as calculated from (7). Through this single reward the RL agent needs to learn to utilize its movements for, distributing the rates during DL, moving during the learning period, and distributing the rates during UL.

C. DEEP Q-LEARNING

Due to the size of MDP, we create an RL agent as a feed-forward neural network (NN), with F input neurons, Y hidden states each with the same number of neurons Z, all using rectified linear (ReLU) activation functions, and an output neuron count of 7. Compared to other state of the art Q-function approximators, our selected NNs provide reasonable accuracy with lower convergence time when compared to, e.g., recurrent NNs (RNNs). When receiving the current state, described with F features as input, the NN agent outputs its evaluation for all seven actions that can be taken. However, the use of NNs in RL tasks may fail to converge especially in problems with complex optimal policy in great state space [42], such as ours. Therefore, we rely on deep RL, using double Q-learning and experience replay methods derived from [43] to bring the problem as close as possible to traditional supervised learning.

Experience replay requires that we store past episodes in a replay buffer. An experience e is defined as a tuple of five elements that occurred during action step t as in $e_t = (s_t, a_t, r_t, s_{t+1}, d_t)$ where: s_t was the starting state of the agent, a_t is the action that the agent took, r_t is the reward that the agent received, s_{t+1} the state at which it arrived, and $d_t = \{0, 1\}$ is an indicator for a terminating action that has 1 if the action finishes the episode or 0 if it leads to another state. This allows the use of mini batches of size β from the stored experiences. However, considering the vast amount of possible trajectories that the drone can try there is still a need to reduce the many unproductive trajectories while exploring the state space. Hence, we also include a secondary experience buffer that only stores the ten best performing episodes. Both buffers are used for training each episode as it was observed that such approach accelerates the convergence while maintaining the exploration.

TABLE 1. Training parameters for RL trajectory optimization.

Label	Definition	Value
γ	Q-learning discount factor	0.9999
α	Learning Rate	0.0001
τ	Soft copy coefficient for double Q-learning	0.005
Y	Number of hidden layers	3
Z	Number of neurons per hidden layer	512
ϵ	Greedy epsilon coefficient	0.5
ψ	Decay factor for epsilon	0.99
β	Number of randomly chosen experiences per batch	256

Finally, for the double-Q-learning RL algorithm, we need to keep two separate agents with the same properties but with different weight values w_{P} and w_{T}. As such they will output a different Q-action function when given the same state. One is used to choose the actions, called a primary model $Q_{\text{P}}(s_t, a_t)$, while the other model evaluates the action during the training, called a target model $Q_{\text{T}}(s_t, a_t)$. Therefore training occurs when taking a batch of experiences e_t from the buffer that is used to update the model as:

$$Q_{\text{P}}^{\text{new}} = (1-\alpha)Q_{\text{P}} + \alpha\big[r_t + (1-d_t)\gamma \max Q_{\text{T}}(s_{t+1}, a)\big], \tag{39}$$

where $\max Q_{\text{T}}(s_{t+1}, a)$ is the action chosen as per the agent, α is the learning rate which was an input to the Adam optimizer [44], and γ is a discount factor that reduces the impact of long term rewards. We implement this with soft updates where instead of waiting several episodes to replace the target model with the primary. The target model receives continuous updates discounted by value τ as in $w_{\text{T}} = w_{\text{T}} \cdot (1-\tau) + w_{\text{P}} \cdot \tau$. Finally, to input new instances in the experience memory, the state space was sampled using the ϵ-greedy strategy. Here, with probability ϵ we take an action uniformly at random from the action space A, while with probability $1-\epsilon$ we act greedily in favor of the agent's decision. After each episode, we lower the ϵ value by multiplying it with a decay coefficient ψ as in $\epsilon^{\text{new}} = \epsilon \cdot \psi$.

D. TRAINING AN RL AGENT FOR TRAJECTORY OPTIMIZATION

In our implementation the drone is trained in an offline manner. In other words, the drone will compute the full trajectory before taking any action in the simulated environment, which is modeled as an MDP. Since we use off-policy learning with sampling, this is easily adaptable to online training, where the ML model would be continuously trainable even during its operation. This is done with scalability in mind as we anticipate that integrated FL implementations require more fine tuned reward systems.

During the training process, RL needed roughly 50-100 episodes of replaying the same scenario solely to understand the situation and converge to stable flying routes that do not incorporate random erratic movements. Once such stability was achieved, the RL agent managed to learn to fly towards

TABLE 2. Testing environment settings [8], [38], [45].

Label	Definition	Value
f_c	Channel Carrier Frequency	5.8 Ghz
W	Channel Bandwidth for each k	20 Mhz
N_0	Noise Spectral Power	-174 dBm/Hz
H	Drone's flying altitude	100 m
$v_{\max}$	Maximum achievable drone speed	10 m/s
Δ	Time Discretization Interval	0.2 s
P_t	Transmission Power	23 dBm
r	Localization Precision Radius	1 m
B	Machine Learning Model Size	2000 Mb
$D_{\max}$	Radius of Coverage Area	260 m
T	Federated Learning Cycle Deadline	30 s
E_r	Antenna Effectiveness	0.6
a	P(LoS) Constant for Suburban topology	4.88
b	P(LoS) Constant for Suburban topology	0.43
η_{LoS}	Added Large Scale for LoS group	0.2
η_{NLoS}	Added Large Scale for NLoS group	24

slow learners and away from learners that received too much attention. After an extensive experimentation, the parameters given in Table 1 resulted in a good learning progress. The results provided in Section V were achieved within 400 episodes of training, which took roughly $80-120$ minutes of training on a GPU accelerated implementation of the script written in Python.

V. SIMULATION RESULTS AND ANALYSIS

The purpose of our simulation is to best evaluate the behaviour of our proposed solutions to the AAS minimization problem that arises in asynchronous FL networks. Moreover, we aim to both validate our framework and decide on an optimization approach that handles the problem based on the values introduced in Section II. Since the performance of the provided trajectories strongly depends on the arrangement of the nodes, both approaches were put against a few handpicked scenarios. We chose this testing approach as aggregation of many randomized runs is unlikely to provide useful results due to the uniqueness of each scenario. Additionally, as our goal is to inspect the usefulness of using trajectory when maximizing AAS, we test on deployments with few nodes. In such scenarios it is easier to sample a challenging scenario for the optimization problems and execute a performance assessment that is directly observable.

In Table 2 we list the parameters used for testing both solutions. The parameters were inspired by object detection models as in IMAGEAI's YOLOv3,[2] being served by a rotary-wing UAV[3] in a suburban environment. We judge the performance of the algorithms by how well they maximize the AAS metric from (7) that is directly dependent on the average learning done by all nodes subtracted by the achieved *staleness* s. As such, the drone trajectory optimization becomes a balancing act of anticipating moving towards and away from specific learners and pre-calculating the possible *staleness* for that learning cycle. The results shown in this section are calculated with a simulation of the non-approximated scenario in Section II, common for both SCP and RL.

2. https://imageai.readthedocs.io/en/latest/detection/
3. https://www.dji.com/dk/matrice100/info#specs

TABLE 3. Coordinates and computational capability for each node k in all three tested scenarios.

	Straight $K=2$			**Hidden** $K=3$			**Forced** $K=3$		
	X_k	Y_k	f_k	X_k	Y_k	f_k	X_k	Y_k	f_k
k=1	60	129	1	10	43	1	0	0	1
k=2	260	0	0.7	29	11	1	1	183	0.7
k=3	N/A	N/A	N/A	169	183	0.7	40	43	0.7

For testing, we assume that the drone should start at the center of the CA, $p_0^d = [0, 0]$. This is done in accord with many works that use a recharging or a battery-swap station [46] at the center of the CA in order to offer uninterrupted, seamless, and standalone service. In our simulated scenarios, we consider the case in which the drone has finished a battery swap from a ground station in the center of the CA, elevated itself to a height H, and initiated its service while it is at a very unbalanced position with regards to the node arrangement.

In this way we test the trajectory optimization approaches in a very challenging environment, where the drone trajectory starts in an unbalanced and therefore unfavorable position with regards to the ground FL network. These simulations cover the first cycle of the drone flight, after the end of which, changes in the area may occur as some learners drop due to lack of data or new learners appearing in other areas of the CA. Therefore, in each cycle the drone will perform swings across the field of learners, until the time comes for the drone to go back to the center, recharge, and get back to orchestrating the FL network.

We tested our approach and evaluated its performance in Table 3. These three scenarios are representative of special case arrangements in which the drone needs to take quick action to best improve the FL's performance for that specific FL cycle. The computational capability f_k for each device in Table 3 is also taken in reference of the task of object detection, where we expect that one epoch should last roughly one second on a computationally powerful node.

A. STRAIGHT TRAJECTORY

The first arrangement, and named Straight, considers only two users $K = 2$, where the drone has a clear path to the slowest learner located in the rightmost corner. If no trajectory is considered, and the service provider is static at the starting location it would achieve an AAS of 3.90 with $s = 5.74$. The SCP solution provides a direct flight towards the slowest learner and achieving an AAS of 7.32 and a *staleness* $s = 3$, outperforming the static implementation by 87.7%. On the other hand, the RL algorithm takes a complicated trajectory, as shown on Fig. 6, and achieves worse results an AAS of 7 with $s = 3.7$, outperforming the static implementation by 79.48%.

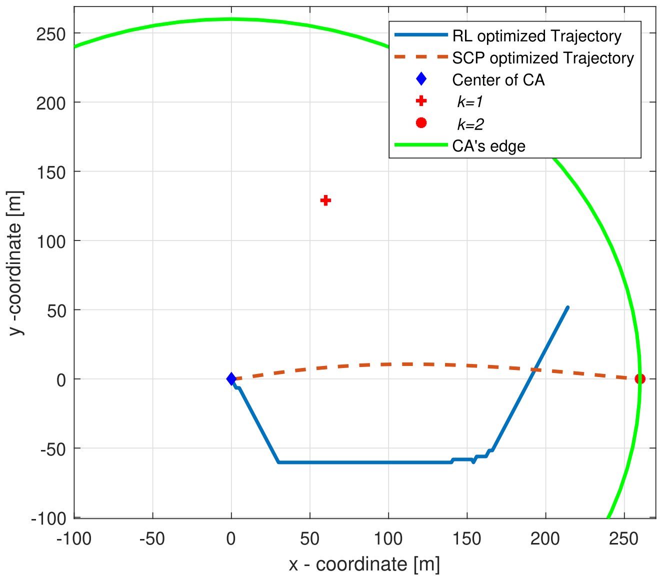

FIGURE 6. Staleness minimization solutions for Straight where: RL yields an AAS of 7 with s=3.7; SCP yields AAS=7.32 with s=3.

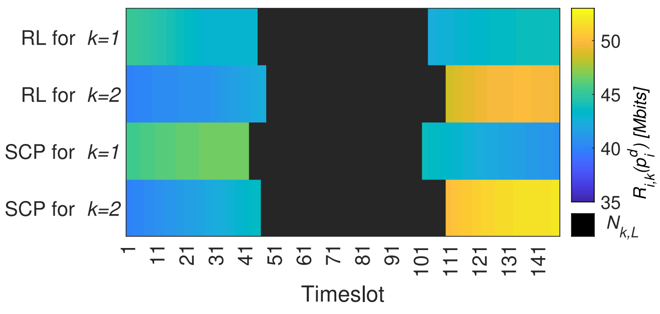

FIGURE 7. A datarate heatmap showcasing DL (left) - Learning (middle) - UL (right) schedule for both RL and SCP approaches for each device in Straight.

Recall that, although no stochastic parameters were involved, the ϵ-greedy approach was used in the RL training for the purpose of exploring the state space. This is significant since it leads the drone to position itself in undesirable locations, at times. In such cases, the agent eventually receives a lower reward even though the trajectory change was caused by a random epsilon event. Thus, RL produces a skewed trajectory in which the drone dramatically evades the good learner $k = 2$ in a behaviour that is most likely due to overstating the importance of *staleness* and the positioning of $k = 2$. RL is outperformed in this simple scenario mostly due to the lack of overfitting mitigation for the Q-learning agent. In future works, we intend to investigate an improved agent with a combination of RNNs and dropout as in [47]. In Fig. 7, we show the chronological progression of the data rates for the ML model transmissions for each link in the Straight scenario. The schedule is clearly visible, starting with the DL phase (colored bars on the left), followed by a period of learning (central black bars), and finalize with the UL phase (colored bars on the right). Therefore, the width of each bar manifests the duration of the phase for that node.

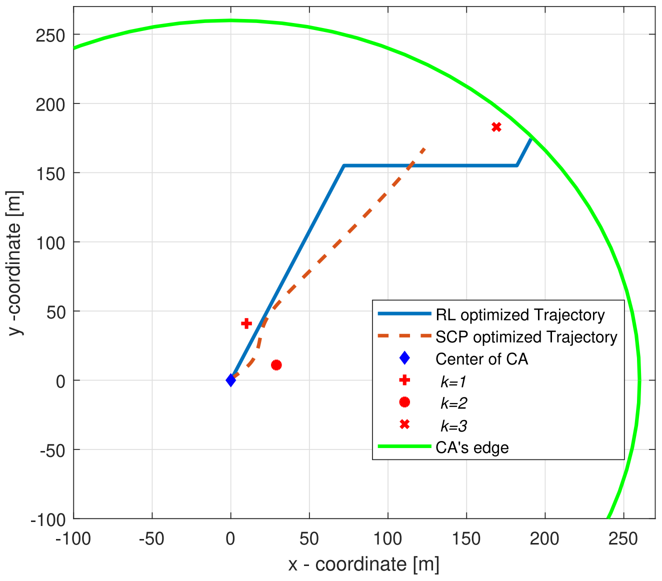

FIGURE 8. Staleness minimization solutions for Hidden where: RL yields an AAS of 7.8 with s=3.7; SCP yields an AAS of 7.1 with s=4.72.

The graph enables us to further analyze the implication of the trajectory, and we notice that the major difference occurs in the DL phase where the RL provided trajectory manages to decrease the rate towards $k = 1$ enough to finish later than $k = 1$ with the SCP trajectory. Since this also comes at a cost to the rate for the straggler node $k = 2$, this measure does not sufficiently improve the staleness value in the RL case so as to outperform the SCP provided solution even though their straggler performance is nearly equal.

B. ONE HIDDEN NODE

The second arrangement, shown in Fig. 8 and named Hidden, has three learners $K = 3$ present in the CA. Here the slowest learner is hidden far away from the center with two well performing nodes in the way of the direct trajectory connecting it. If static at the starting location, the orchestrator would achieve an AAS of 4.52 with $s = 7.46$. Both algorithms show prowess to discover the policy of breaching the barrier of fast learners to get to the slowest, hence providing an adequate solution. However, the RL algorithm does this much more efficiently with an AAS of 7.8 with $s = 3.7$, outperforming the static implementation by 72.56%. RL outperforms its SCP counterpart that achieved an AAS of 7.1 with $s = 4.72$, outperforming the static implementation by 57.08%. In both SCP and RL implementations, the drone is trying to leave the barrier of good learners as fast as possible and move towards the neglected user to match the rate it gave to the fast learners in the DL phase. In this testing scenario, using unapproximated rate goes massively in the benefit of discovering good positions with high precision for the RL approach. Additionally, in Fig. 9, we show the evolution of the transmission schedule for the Hidden scenario and highlight the better performance of the RL implementation particularly when dealing with the slow learner $k = 3$.

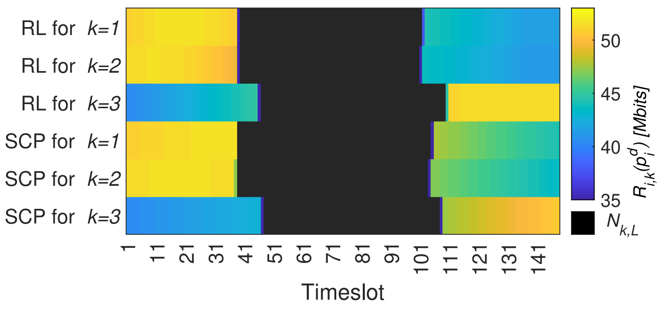

FIGURE 9. A datarate heatmap showcasing DL (left) - Learning (middle) - UL (right) schedule for both RL and SCP approaches for each device in Hidden.

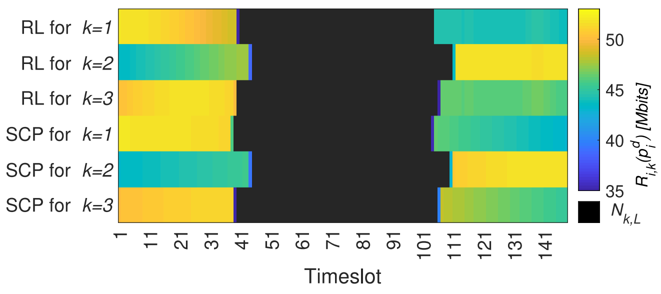

FIGURE 11. A datarate heatmap showcasing DL (left) - Learning (middle) - UL (right) schedule for both RL and SCP approaches for each device in Forced.

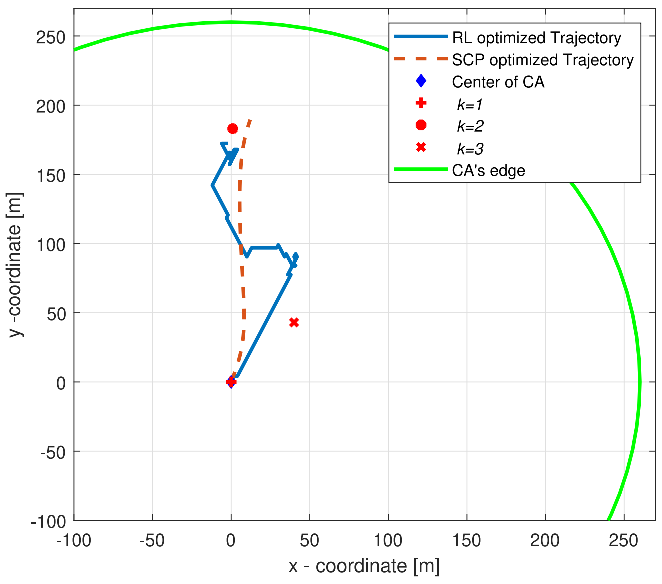

FIGURE 10. Staleness minimization solutions for Forced where: RL yields an AAS of 7.2 with s=3.48; SCP yields an AAS of 6.74 with s=4.02.

Here, the RL guided drone travels very close to $k = 3$ during the final UL phase and, thus, it achieves near maximum rates and transfers ≈ 52 Mbits each timeslot.

An important note is that during RL training, it was observed that the RL agent would sometimes try a policy of flying away from all nodes in an effort to reduce the performance of nodes $k = 1$ and $k = 2$. This signifies the superiority of using our AAS metric as opposed to solely relying on s as an optimization metric since AAS "anchors" the drone from moving away from all learners. If we only relied on *staleness* as an independent optimization metric it would reflect heavily in the total amount of computation work done by the FL nodes, therefore slow down FL convergence.

C. FORCED DEPARTURE

Next, in Fig. 10, we consider a third scenario named Forced that has one well performing node directly in the center of the CA and two slow learners scattered at two positions away from the center. This scenario forces the drone orchestrator to quickly move out from its starting position to the closest and fastest learner, to serve a more balanced role away and towards the two other nodes. Here a static drone fixed at the starting location would achieve an AAS of 4.22 with $s = 6.62$.

This scenario showcases how the proposed RL algorithm allows the drone to find a high-performing trajectory that yields an AAS of 7.2 with $s = 3.48$, thus significantly outperforming the static implementation by 70.61%. In contrast, here, the SCP algorithm achieved an AAS of 6.74 with $s = 4.02$, outperforming the static implementation by only 59.71%. As the drone shuffles around after reaching some satisfactory distance from the good node, it is apparent that RL-guided trajectories could incorporate many erratic and unnecessary movements. This behavior of RL is clearly visible in this scenario due to the relative closeness of all three nodes next to the starting location. Finally, in Fig. 11 we can observe that the performance of both RL and SCP is very similar in treating the problem of maximizing AAS. Nonetheless, it is noticeable that RL does well at compensating the most disadvantaged node $k = 2$, particularly in the UL phase.

D. KEY TAKEAWAYS

In a nutshell, the RL implementation is bound to be superior in comparison to the SCP approach due to the granularity of each action it can produce by its progressive decisions. Additionally, as RL needs no approximations for the environmental parameters, it is capable of discovering true optimums, if given the time and well done exploration, overfitting avoidance, and good hyper-parameter adjustment. However, to advance the RL implementation in a realistic environment, it is also beneficial to perform online learning. This would come with great energy requirements as the drone would have to dedicate a lot of its processing capabilities to improve the implementation (although some of this could be alleviated through the offline training phase). As such, simplified approximations such as the SCP approach may be more feasible to implement in first deployments of drone orchestrators. We are optimistic that later designs of RL implementations that are guided by the SCP algorithm will provide superior performance. This combination

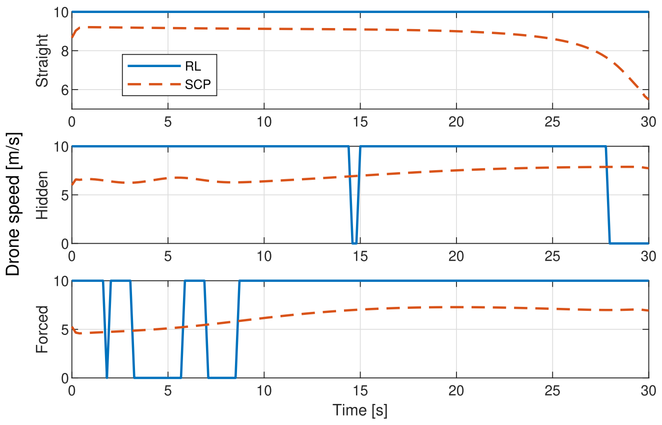

FIGURE 12. **The speed distribution of the drone across time for both RL and SCP guided trajectories, for all three scenarios.**

with offline and online RL training, should be adept in finding near optimal trajectories in every stochastic, chaotic and energy limited implementations.

An important takeaway from all three scenarios is that SCP generally reacts a lot slower to the distance discrepancies mainly due to the approximations needed for converting the problem into a convex one. However, the RL approach yields many unnecessary movements that may have negative effects on the drone battery life and therefore its flight duration. For example, Fig. 12 shows that RL provided trajectories keep the drone at full speed for the majority of the time. This, combined with our previous result analysis for Fig. 10, we can conclude that often times this is unnecessary for the AAS performance. The common way to address this in RL is to assign negative rewards for drone's movements. This will need to be carefully designed in order to not impair the performance of the FL network, given the long-term utility of the application.

VI. CONCLUSION

In this paper, we have considered a drone equipped with a wireless interface in the role of an orchestrator in an FL implementation where it coordinates the transmission and learning by only adjusting its flying trajectory, and therefore, the horizontal distance to each node. Considering the total amount of learning performed across all nodes and the learning discrepancies between them as an optimization criterion, two trajectory optimization approaches were compared: deep RL and SCP. From our analysis of their potential in maximizing the combined performance metric, we have concluded that RL has shown its suitability and general superiority in solving the task of trajectory optimization for lowering *staleness* in FL networks. Nonetheless, both solutions for the drone-orchestrated FL concept outperform a static implementation with improvements in the range of 57% to 87.7%. All in all, SCP approaches are simpler and reliably provide decent performance in all scenarios without the need of hardware accelerated computing. As such, both approaches are important in transitioning towards superior FL implementations for scattered networks. However, RL implementations are necessary when encountering realistic wireless channel, weather and user mobility conditions. This inspires future works where an RL agent goes trough the process of pre-training with SCP approximated trajectories, and is then transferred for learning in the real world environments.

REFERENCES

[1] A. Fotouhi *et al.*, "Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3417–3442, 4th Quart., 2019.

[2] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, 3rd Quart., 2019.

[3] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May/Jun. 2019.

[4] Z. Yang, M. Chen, K.-K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6G: Applications, challenges, and opportunities," 2021. [Online]. Available: arXiv:2101.01338.

[5] A. Fotouhi, M. Ding, and M. Hassan, "Dynamic base station repositioning to improve performance of drone small cells," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[6] B. Galkin, J. Kibiłda, and L. A. DaSilva, "A stochastic model for UAV networks positioned above demand hotspots in urban environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6985–6996, Jul. 2019.

[7] M. Mozaffari, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, "Beyond 5G with UAVs: Foundations of a 3D wireless cellular network," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 357–372, Jan. 2019.

[8] I. Donevski and J. J. Nielsen, "Dynamic standalone drone-mounted small cells," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Dubrovnik, Croatia, Sep. 2020, pp. 342–347.

[9] 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.

[10] N. Babu, K. Ntougias, C. B. Papadias, and P. Popovski, "Energy efficient altitude optimization of an aerial access point," in *Proc. IEEE 31st Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, London, U.K., Sep. 2020, pp. 1–7.

[11] N. Babu, C. B. Papadias, and P. Popovski, "Energy-efficient 3-D deployment of aerial access points in a UAV communication system," *IEEE Commun. Lett.*, vol. 24, no. 12, pp. 2883–2887, Dec. 2020.

[12] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.

[13] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 328–331, Jun. 2018.

[14] C. You and R. Zhang, "3D trajectory optimization in Rician fading for UAV-enabled data harvesting," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3192–3207, Jun. 2019.

[15] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2125–2140, Apr. 2019.

[16] Y. Chen, X. Lin, T. Khan, and M. Mozaffari, "Efficient drone mobility support using reinforcement learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Seoul, South Korea, May 2020, pp. 1–6.

[17] H. Bayerlein, P. De Kerret, and D. Gesbert, "Trajectory optimization for autonomous flying base station via reinforcement learning," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2018, pp. 1–5.

[18] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.
[19] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.
[20] C. Pradhan, A. Li, C. She, Y. Li, and B. Vucetic, "Computation offloading for IoT in C-RAN: Optimization and deep learning," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4565–4579, Jul. 2020.
[21] W. Y. B. Lim *et al.*, "Towards federated learning in UAV-enabled Internet of vehicles: A multi-dimensional contract-matching approach," 2020. [Online]. Available: arXiv:2004.03877.
[22] J. S. Ng *et al.*, "Joint auction-coalition formation framework for communication-efficient federated learning in UAV-enabled Internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2326–2344, Apr. 2021.
[23] H. Zhang and L. Hanzo, "Federated learning assisted multi-UAV networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 14104–14109, Nov. 2020.
[24] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad, and M. Bennis, "Federated learning in the sky: Joint power allocation and scheduling with UAV swarms," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.
[25] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Nov. 2021.
[26] M. Bergerman, J. Billingsley, J. Reid, and E. van Henten, "Robotics in agriculture and forestry," in *Handbook of Robotics*. Heidelberg, Germany: Springer-Verlag, 2016, pp. 1463–1492.
[27] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," Oct. 2016. [Online]. Available: arXiv:1610.05492.
[28] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," Dec. 2018. [Online]. Available: arXiv:1812.06127.
[29] S. J. Kim and G. J. Lim, "Drone-aided border surveillance with an electrification line battery charging system," *J. Intell. Robot. Syst.*, vol. 92, nos. 3–4, pp. 657–670, Dec. 2018.
[30] W. Zhang, S. Gupta, X. Lian, and J. Liu, "Staleness-aware async-SGD for distributed deep learning," Nov. 2015. [Online]. Available: arXiv:1511.05950.
[31] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," Mar. 2019. [Online]. Available: arXiv:1903.03934.
[32] U. Mohammad and S. Sorour, "Adaptive task allocation for asynchronous federated mobile edge learning," May 2019. [Online]. Available: arXiv:1905.01656.
[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, Oct. 2018.
[34] I. Donevski, J. J. Nielsen, and P. Popovski, "Standalone deployment of a dynamic drone cell for wireless connectivity of two services," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Nanjing, China, Apr. 2021.
[35] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," Feb. 2019. [Online]. Available: arXiv:1902.00146.
[36] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.
[37] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.
[38] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *Proc. IEEE Global Commun. Conf.*, Austin, TX, USA, Oct. 2014, pp. 2898–2904.
[39] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.
[40] T. D. Quoc and M. Diehl, "Sequential convex programming methods for solving nonlinear optimization problems with DC constraints," Jul. 2011. [Online]. Available: arXiv:1107.5841.
[41] M. Grant, S. Boyd, and Y. Ye. (2009). *CVX: MATLAB Software for Disciplined Convex Programming*. [Online]. Available: http://cvxr.com/cvx/
[42] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[43] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," Sep. 2015. [Online]. Available: arXiv:1509.06461.
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014. [Online]. Available: arXiv:1412.6980.
[45] C. She, C. Liu, T. Q. S. Quek, C. Yang, and Y. Li, "Ultra-reliable and low-latency communications in unmanned aerial vehicle communication systems," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3768–3781, May 2019.
[46] K. Fujii, K. Higuchi, and J. Rekimoto, "Endless flyer: A continuous flying drone with automatic battery replacement," in *Proc. IEEE 10th Int. Conf. Ubiquitous Intell. Comput.*, Vietri sul Mare, Italy, Dec. 2013, pp. 216–223.
[47] I. Donevski, G. Vallero, and M. A. Marsan, "Neural networks for cellular base station switching," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops*, Paris, France, Apr. 2019, pp. 738–743.

IGOR DONEVSKI (Graduate Student Member, IEEE) received the bachelor's degree in telecommunications and information engineering from the University of Ss. Cyril and Methodius, North Macedonia, in 2016, and the M.Sc. degree in communications and computer networks from the Politecnico di Torino, Italy, in 2018. He was simultaneously enrolled in the honors program of Alta Scuola Politecnica and received a double degrees with Politecnico di Milano. He is a Marie-Curie Early Stage Researcher for the EU H2020 ITN Project PAINLESS and a Ph.D. Fellow with Aalborg University, Denmark. In 2015, he completed a two-month visit with Northwestern University, USA, under the supervision of Prof. Berry, where he worked on energy constrained communications. His general interests reside in wireless communications, machine learning, and energy efficient communication systems.

NITHIN BABU (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from CUSAT, India, in 2013, and the M.Tech. degree in communication systems engineering from Indian Institute of Technology Patna, Patna, India, in 2016. He is currently pursuing the Ph.D. degree with Aalborg University, Denmark. He is an early stage Researcher with Alba, The American College of Greece for the EU H2020 ITN Project PAINLESS. During his M.Tech., he was a DAAD Exchange Student with Vodafone Chair, TU Dresden. His research interests include UAV communication, LoS-MIMO systems, and mm-wave communication. He was awarded as the Best Student in order of merit.

JIMMY JESSEN NIELSEN (Senior Member, IEEE) received the Ph.D. degree in wireless communications from Aalborg University, Denmark, in 2011, where he is an Associate Professor of Wireless Connectivity with the Department of Electronic Systems. He has authored and coauthored more than 40 publications in conferences, journals, and books, and has served as reviewer for various IEEE journals and conferences. His research interests include modeling and performance analysis of ultra-reliable communication and low latency communication (URLLC), Internet of Things, and drone communications. He was a Publications Chair and a Co-Organizer of the IEEE SmartGridComm 2018 Conference, and has co-organized several workshops and special sessions, latest in ISWCS 2018, WCNC 2019, and ISWCS 2019.

PETAR POPOVSKI (Fellow, IEEE) received the Dipl.-Ing. and M.Sc. degrees in communication engineering from the University of Ss. Cyril and Methodius, Skopje, and the Ph.D. degree from Aalborg University in 2005. He is a Professor with Aalborg University, where he heads the section on Connectivity. He has authored the book *Wireless Connectivity: An Intuitive and Fundamental Guide* (Wiley, 2020). His research interests are in the area of wireless communication and communication theory. He received an ERC Consolidator Grant in 2015, the Danish Elite Researcher Award in 2016, the IEEE Fred W. Ellersick Prize in 2016, the IEEE Stephen O. Rice Prize in 2018, the Technical Achievement Award from the IEEE Technical Committee on Smart Grid Communications in 2019, and the Danish Telecommunication Prize in 2020. He is currently an Area Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He was the General Chair for IEEE SmartGridComm 2018 and IEEE Communication Theory Workshop 2019. He is a member of Large at the Board of Governors in IEEE Communication Society, a Vice-Chair of the IEEE Communication Theory Technical Committee, and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.

WALID SAAD (Fellow, IEEE) received the Ph.D. degree from the University of Oslo in 2010. He is currently a Professor with the Department of Electrical and Computer Engineering, Virginia Tech, where he leads the Network sciEnce, Wireless, and Security (NEWS) Laboratory. His research interests include wireless networks, machine learning, game theory, security, unmanned aerial vehicles, cyber–physical systems, and network science. He is also a 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 in 2015. He was the author/coauthor of ten conference best paper awards at WiOpt in 2009, ICIMP in 2010, IEEE WCNC in 2012, IEEE PIMRC in 2015, IEEE SmartGridComm in 2015, EuCNC in 2017, IEEE GLOBECOM in 2018, IFIP NTMS in 2019, IEEE ICC in 2020, and IEEE GLOBECOM in 2020. He is a recipient of the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, 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. He was also a coauthor of the 2019 IEEE Communications Society Young Author Best Paper. He received the Dean's Award for Research Excellence from Virginia Tech in 2019. From 2015 to 2017, he was named the Stephen O. Lane Junior Faculty Fellow at Virginia Tech, and in 2017 he was named College of Engineering Faculty Fellow. He currently serves as an Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING and the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. He is an Editor-at-Large for the IEEE TRANSACTIONS ON COMMUNICATIONS. He is a Fellow of the IEEE Distinguished Lecturer.