Distributed Multi-agent Meta Learning for Trajectory Design in Wireless Drone Networks

Ye Hu, Mingzhe Chen, *Member, IEEE*, Walid Saad, *Fellow, IEEE*, H. Vincent Poor, *Life Fellow, IEEE*, and Shuguang Cui, *Fellow, IEEE*

***Abstract*—In this paper, the problem of the trajectory design for a group of energy-constrained drones operating in dynamic wireless network environments is studied. In the considered model, a team of drone base stations (DBSs) is dispatched to cooperatively serve clusters of ground users that have dynamic and unpredictable uplink access demands. In this scenario, the DBSs must cooperatively navigate in the considered area to maximize coverage of the dynamic requests of the ground users. This trajectory design problem is posed as an optimization framework whose goal is to find optimal trajectories that maximize the fraction of users served by all DBSs. To find an optimal solution for this non-convex optimization problem under unpredictable environments, a value decomposition based reinforcement learning (VD-RL) solution coupled with a meta-training mechanism is proposed. This algorithm allows the DBSs to dynamically learn their trajectories while generalizing their learning to unseen environments. Analytical results show that, the proposed VD-RL algorithm is guaranteed to converge to a local optimal solution of the non-convex optimization problem. Simulation results show that, even without meta-training, the proposed VD-RL algorithm can achieve a 53.2% improvement of the service coverage and a 30.6% improvement in terms of the convergence speed, compared to baseline multi-agent algorithms. Meanwhile, the use of meta-training mechanism improves the convergence speed of the VD-RL algorithm by up to 53.8% when the DBSs must deal with a previously unseen task.**

***Index Terms*—Drones, network optimization, multi-agent reinforcement learning, meta-learning.**

I. Introduction

Aerial wireless communication platforms carried by drones can provide a cost-effective, flexible approach to boost the coverage and capacity of future wireless networks [2]–[4]. However, effectively deploying a group of drone base stations (DBSs) for providing timely on demand wireless connectivity to ground users in dynamic wireless environments is still an important open problem. In particular, designing trajectories for a group of independent DBSs is challenging particularly when the DBSs only have limited information on the wireless requests of the ground users, which are often highly unpredictable and dynamic.

A. Related Works

The existing literature in [5]–[18] studied a number of problems related to trajectory design for drone-based wireless networks. The work in [5] studies the drone trajectory optimization problem by jointly considering both the drone's communication throughput and its energy consumption. The authors in [6] design the trajectory of a solar-powered DBS to enhance its wireless communication performance. In [7], the problem of trajectory design and user association in a multi-drone communication system is solved with a block coordinate descent solution. The authors in [8] propose a dynamic trajectory control algorithm to improve the communication performance of the DBSs. The authors in [9] optimize time allocation, reflection coefficient adjustment, and DBS trajectory in backscatter communication networks. Despite their promising results, these existing works [5]–[9] do not consider practical DBS-assisted wireless networks in which the ground user requests for wireless service follow unpredictable patterns. Indeed, the optimization based solutions in [5]–[9] are not suitable to design DBS trajectories when the user requests are unknown and unforeseeable.

Recently, there has been significant interest in realizing intelligent control of drones in dynamic networking environments, using machine learning tools for trajectory design [10]–[18]. In [10], the problem of real-time dynamic maneuver design at a data collecting DBS is studied and solved using reinforcement learning (RL). The work in [11] employs two powerful deep neural networks to intelligently guide an energy-limited DBS under environmental dynamics. In [12], the authors develop an RL algorithm that enables a drone to act as a data-collection relay with the goal of maximize information freshness. However, the work in [12] is restricted to the case of a single drone. For intelligently controlling multi-drone systems, the authors in [13] design interference-aware paths for a group of cellular-connected drones by using

This research was supported by the U.S. National Science Foundation (NSF) under Grants CNS-1909372, CNS-1836802, and CCF-1908308. It was also supported in part by the National Key R&D Program of China with grant No. 2018YFB1800800, by the Key Area R&D Program of Guangdong Province with grant No. 2018B030338001, by Shenzhen Outstanding Talents Training Fund, and by Guangdong Research Project No. 2017ZT07X152. A preliminary version of this work will be presented at the IEEE Global Communications Conference [1].

Y. Hu, and W. Saad are with the Wireless@VT, Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA, 24061, Emails: yeh17@vt.edu, walids@vt.edu. W. Saad is also with the Department of Computer Science and Engineering, Kyung Hee University, South Korea.

M. Chen is with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, 08544, and the Future Network of Intelligence Institute, Chinese University of Hong Kong, Shenzhen, China, 518172, Email: mingzhec@princeton.edu.

H. V. Poor is with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, 08544, Emails: poor@princeton.edu.

S. Cui is currently with the Shenzhen Research Institute of Big Data and Future Network of Intelligence Institute (FNii), the Chinese University of Hong Kong, Shenzhen, China, 518172, Email: e-mail: shuguangcui@cuhk.edu.cn.

a multi-agent reinforcement learning (MARL) solution that relies on a deep echo state network (ESN) architecture. The work in [14] proposes a deep MARL architecture that directs drones within a continuous set of locations to accomplish time-sensitive sensing tasks. Meanwhile, in [15], the authors develop a MARL framework to allow drones dynamically manage resources according to their local observations. The authors in [16] propose an ESN based learning architecture to predict the users' mobility patterns and, then, realize an optimal deployment of a group of DBSs. The works in [17] and [18] propose distributed multi-agent algorithms that allow a group of agents to update their individual strategies considering the team benefits. However, most of the existing MARL solutions such as those in [13], and [16]–[18] require DBSs to share their states and actions while searching for the optimal strategies. These traditional RL solutions have high complexity, as they solve multi-agent problems by updating strategies based on the entire set of agents' actions and strategies whose dimension increases exponentially with the number of agents. Meanwhile, the MARL solutions in [14] and [15] allow the agents to search strategies independently based on their own actions and states. However, using these RL solutions, the DBSs cannot optimize the sum utilities of all DBSs, and thus, cannot maximize the overall coverage of the ground users, since the DBSs are optimizing their individual utilities. In addition, traditional RL solutions such as those used in [13]–[18] cannot efficiently adapt the trajectories of the DBSs to unseen environments, as they are often overfitted to their training tasks. This is because the hyper-parameters, exploration strategies, and initializations of traditional RL algorithms are manually adjusted for fitting the training tasks. Once the agent faces an unseen task, manually adjusted RL algorithms may not converge to the optimal solution and, even if they do, the convergence speed will be very slow. As a result, the traditional RL algorithms in [10]–[18] cannot effectively find optimal DBS trajectories in unseen environments. Finally, we note that in [1], we studied the problem of trajectory design for a single DBS operating in a dynamic environment using meta-learning. However, this prior work relies on a simple algorithm that cannot be scaled to larger networks.

B. Contributions

The main contribution of this paper is a novel distributed framework for designing the trajectories of a group of cooperative DBSs in unpredictable, dynamic environments. To our best knowledge, *this is the first work that designs trajectories for a team of DBSs in unpredictable, dynamic environments using a multi-agent meta reinforcement learning solution.* In brief, our key contributions include:

- We consider a practical drone-aided wireless system in which a team of DBSs cooperatively navigate in an area, under strict energy constraints and limited information on surrounding environments, with the goal of providing uplink wireless connectivity to ground users. The DBSs can provide on-demand coverage to the ground users while adapting their trajectories to those users' unpredictable access requests. We formulate this trajectory design problem as an optimization framework whose structure is shown to be non-convex. To solve this problem, the "myopic" DBSs, which have only limited access to the information of ground users, seek to find trajectories that maximize the expected portion of served users – a wireless coverage metric that we use as the team utility of the group of DBSs.
- To solve the formulated trajectory optimization problem, we propose a novel, distributed, value decomposition reinforcement learning (VD-RL) algorithm. This algorithm is shown to reach a local optimal solution of the studied non-convex problem without requiring the DBSs to share their actions, states, or strategies. This makes the proposed VD-RL algorithm much less complex than traditional distributed MARL algorithms (e.g., those in [13], and [16]–[18]), as the DBSs can update their strategies based on their own low-dimensional actions and states. The proposed VD-RL algorithm allows the DBSs to independently select strategies that maximize the team utility by decomposing and attributing this team utility to each DBS. Thus, with the proposed VD-RL algorithm, the DBSs can find a local optimal solution, which yields a much higher team utility, compared to the one that can be achieved by existing MARL solutions in [14] and [15].
- To improve the convergence speed of VD-RL for unseen environments, we propose a meta training mechanism that uses an optimization based solution to meta train the VD-RL algorithm. In particular, the proposed meta training mechanism seeks to find proper policy and value function initializations that estimate all possible user request patterns for the VD-RL algorithm. Using this meta-learning mechanism, the VD-RL solution can quickly converge to a team optimal strategy when faced with an unseen task. The proposed optimization based meta training mechanism has a lower complexity compared to the meta training solutions in [19] and [20], because it does not require additional neural networks for meta-learning. Using the proposed approach, the DBSs can be meta-trained independently with their own actions and states. In particular, the meta-learning approach helps DBSs to cope with various tasks instead of a single sampled task as in [21], by using a meta-trained initialization.

Simulation results show that proposed VD-RL algorithm can improve the service coverage and convergence speed by up to 53.2% and 30.6%, compared to traditional multi-agent algorithms. The results also show that using the proposed meta training mechanism, our VD-RL algorithm can find optimal trajectories in an unseen environment with a 53.8% faster convergence speed, and a 5.6% better coverage, compared to the VD-RL algorithm without meta-learning.

The rest of this paper is organized as follows. The system model and problem formulation are described in Section II. In Section III, the proposed algorithm is developed and discussed. In Section IV, simulation results are presented.

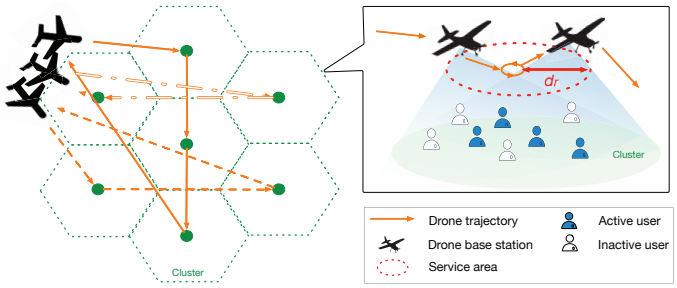


Fig. 1. Topology of our considered network. In this model, the DBSs travel across the clusters in SLF, hover over each cluster with SCF, and keep serving their associated users within each cluster.

Finally, conclusions are drawn in Section V.

II. System Model and Problem Formulation

Consider a geographical area within which a set $\mathcal{U}$ of U randomly deployed terrestrial users request uplink data service. A set $\mathcal{N}$ of N fixed-wing DBSs are dispatched to satisfy the uplink access requests of those ground users, as shown in Fig. 1. In the considered area, an user that requests data service is called an *active user*, otherwise, it will be an *inactive user*. We assume the users to be separated into different groups, each of which is called a *cluster*. Note that, a "cluster" is defined as an area that falls within the service coverage of only one DBS. We also assume that, at any given time, each cluster will be served by a single DBS. The set of such clusters is denoted as $\mathcal{C}$. The DBSs will travel across the clusters in a steady straight-and-level flight (SLF), and hover over each cluster with a steady circular flight (SCF), at a constant speed V_s [5]. Each DBS $n \in \mathcal{N}$ flies at its own constant altitude [1] H_n to avoid collision with other DBSs. All the DBSs must return to their original location O within a time period T for battery charging. Moreover, all DBSs are assumed to have the same battery capacity. The trajectory that records DBS n's movement within the time period T is represented by a vector $\boldsymbol{\xi}_n = [\xi_{n,1}, \xi_{n,2}, \ldots, \xi_{n,K}]^\top$, with $\xi_{n,k} \in \mathcal{C} \cup \{O\}$ being the k-th cluster that DBS n serves, or the initial location O that DBS n returns to after serving the users, and K being the maximum number of locations each DBS can fly across under its energy constraints. In other words, each DBS n flies across no more than K different locations, each of which is called one *step* on the trajectory of DBS n, the set of such steps is denoted as $\mathcal{K}$. For example, $\boldsymbol{\xi}_n = [\xi_{n,1}, O, \ldots, O]$, with $\xi_{n,1} \in \mathcal{C}$, implies that DBS n serves cluster $\xi_{n,1}$ and then flies back to the initial location O, while $\boldsymbol{\xi}_n = [\xi_{n,1}, \xi_{n,2}, \ldots, \xi_{n,K-1}, O]$, with $\xi_{n,k} \in \mathcal{C}$, implies that DBS n serves clusters $\xi_{n,1}, \xi_{n,2}, \ldots, \xi_{n,K-1}$, and then flies back to the initial location O.

A. Communication Performance Analysis

In our network, the users adopt an orthogonal frequency division multiple access (OFDMA) technique and transmit data over a set of uplink *resource blocks* (RBs) [22]. Each dispatched DBS will arbitrarily allocate one RB to each one of its associated users within a cluster. We assume that each DBS can keep serving its associated users within a d_r-meter radius over each cluster as shown in Fig. 1. The area within this range is called a *service area*. Also, user u is assumed to request a total of b_u data (in bits) at time epoch t_u. The DBSs must satisfy these requests within their battery capacities. Let $\boldsymbol{b} = [b_1, \ldots, b_U]$ and $\boldsymbol{t} = [t_1, \ldots, t_U]$ be, respectively, the vector of quantity and occurrence of the users' access request in the network. The quantity b_u and active time t_u are assumed to be independent random variables that follow unknown distributions. Hereinafter, we denote $\boldsymbol{z} = [\boldsymbol{b}, \boldsymbol{t}]$ as one *realization* of the users' access request within duration T. In this model, the deployed DBSs are *myopic*, that is, they only know the access quantities and active time of the users that they are currently serving. The path loss (in dB) of the line-of-sight (LoS) and non-line-of-sight (NLoS) air-to-ground communication links between DBS n to user u are given by the popular air-to-ground model in [23]

$$h_{u,n}^{\text{LoS}} = 20\log\left(\frac{4\pi f_c d_{u,n}}{c}\right) + \varsigma_{u,n}^{\text{LoS}},$$
$$h_{u,n}^{\text{NLoS}} = 20\log\left(\frac{4\pi f_c d_{u,n}}{c}\right) + \varsigma_{u,n}^{\text{NLoS}}, \quad (1)$$

where f_c is the carrier frequency of the communication link between DBS n and user u, $d_{u,n}$ is the distance between user u and DBS n, and c is the speed of light. $\varsigma_{u,n}^{\text{LoS}}$ and $\varsigma_{u,n}^{\text{NLoS}}$ are, respectively, the additional path losses at the LoS and NLoS air-to-ground links between DBS n to user u. The value of $\varsigma_{u,n}^{\text{LoS}}$ and $\varsigma_{u,n}^{\text{NLoS}}$ follow Gaussian distributions with different parameters $(\mu_{\text{LoS}}, \delta_{\text{LoS}}^2)$ and $(\mu_{\text{LoS}}, \delta_{\text{LoS}}^2)$, respectively. Note that the path loss values between DBS n and user u are considered to be stable with DBS n's movement, as the distance between DBS n and user u will only experience small changes when DBS n flies within the service area. The signal-to-noise ratio (SNR) at the LoS and NLoS links between DBS n and user u, will thus be

$$\gamma_{u,n}^{\text{LoS}} = \frac{P}{N_0 B 10^{\frac{h_{u,n}^{\text{LoS}}}{20}}},$$
$$\gamma_{u,n}^{\text{NLoS}} = \frac{P}{N_0 B 10^{\frac{h_{u,n}^{\text{NLoS}}}{20}}}, \quad (2)$$

where P represents the transmit power of user u, which is assumed to be equal for all users. N_0 is the noise power spectral density, and B is the RB bandwidth (equal for all RBs). In order to avoid LoS interference to ground links and because the number of DBSs in real systems will be small, we assume that each DBS will use its own dedicated frequency band. The data rate at the link between DBS n and user u will then be $c_{u,n} = \beta_{u,n}^{\text{LoS}} B \log\left(1 + \gamma_{u,n}^{\text{LoS}}\right) + \beta_{u,n}^{\text{NLoS}} B \log\left(1 + \gamma_{u,n}^{\text{NLoS}}\right)$, where $\beta_{u,n}^{\text{LoS}} = \left[1 + \varphi \exp\left(-\phi \frac{180}{\pi} \theta_{n,u} + \varphi\phi\right)\right]^{-1}$ is the probability of having a LoS link between DBS n and user u, $\beta_{u,n}^{\text{NLoS}} = 1 - \beta^{\text{LoS}}$ is the probability of having a NLoS link between DBS n and user u. Here, φ and ϕ are constant values that depend

[1] Here, the differences in the altitudes between the DBSs are assumed to be considerably small when compared to the DBSs' dedicated altitudes H_n. Thus, the altitude differences only have limited impact on the DBSs' communication performance, including data rates, as well as service coverage.

on the studied communication environments, while $\theta_{n,u}$ is the elevation angle between DBS n and user u.

B. Utility Function Model

In the studied scenario, the goal of the dispatched DBSs is to cover all access requests from ground users. In such a case, the utility of each DBS is defined as the *successful service rate*, which captures the fraction of users being served by a given DBS in a given time period. Note that, when DBS n arrives at a cluster, it will only serve the user requests that were not served by the DBSs that arrived at this cluster before DBS n. For the special case in which multiple DBSs arrive at a cluster at the same time, only one DBS will serve the entire cluster, and the remaining DBSs will directly proceed toward other clusters. Note that once a DBS finds out (with their limited situational awareness) that another DBS is hovering on the service area of a cluster, it knows that this cluster is being served, and it will leave this cluster. Thus, the successful service rate achieved at DBS n by serving cluster $\xi_{n,k}$ is given by

$$\mu_{n,k}\left(\boldsymbol{\xi}\right)=\frac{\sum_{u\in\mathcal{U}}\mathbb{1}_{\{u\in\mathcal{U}_{n,k},T-\tau_n^*\leq t_u\leq T-\tau_{n,k}\}}}{\sum_{u\in\mathcal{U}}\mathbb{1}_{\{0\leq t_u\leq T\}}},\tag{3}$$

where $\boldsymbol{\xi}=[\boldsymbol{\xi}_1,\boldsymbol{\xi}_2,\ldots,\boldsymbol{\xi}_N]$ is the matrix of trajectories of the DBSs, $\mathcal{U}_{n,k}$ is the set of active users in cluster $\xi_{n,k}$. $\tau_{n,k}$ is the time duration that DBS n is allowed to keep flying with its remaining energy level, after successfully serving cluster $\xi_{n,k}$[2]. Note that, hereinafter, the time duration $\tau_{n,k}$ is called the *available service time* of DBS n at step k. In (3), $\mathbb{1}_{\{x\}}=1$ when x is true, otherwise, $\mathbb{1}_{\{x\}}=0$. Here, $\sum_{u\in\mathcal{U}}\mathbb{1}_{\{0\leq t_u\leq T\}}$ is the number of active users within the studied time duration, and $\sum_{u\in\mathcal{U}}\mathbb{1}_{\{u\in\mathcal{U}_{n,k},T-\tau_n^*\leq t_u\leq T-\tau_{n,k}\}}$ is the number of active users served by DBS n in cluster $\xi_{n,k}$. $\tau_n^*=\min_{n'\in\mathcal{N}_n}\tau_{n',k'}$ is the available service time of the last DBS that arrived at cluster $\xi_{n,k}$ before DBS n. $T-\tau_n^*\leq t_u\leq T-\tau_{n,k}$ represents that user u requests data before DBS n's arrival, and have not served by any other DBSs, as shown in Fig. 2. Here, $\mathcal{N}_n=\{n'\,|n'\in\mathcal{N}\setminus n,\xi_{n',k'}=\xi_{n,k},\tau_{n',k'}\geq\tau_{n,k}\}$ is the set of DBSs that arrive at cluster $\xi_{n,k}$ before DBS n, where $\xi_{n',k'}=\xi_{n,k}$ implies that cluster $\xi_{n,k}$ is also the k'-th cluster served by DBS n'. Moreover, DBS n's available service time upon its arrival at cluster $\xi_{n,k}$ on trajectory $\boldsymbol{\xi}_n$ is given by $\tau_{n,k}=T-\sum_{\kappa=0}^{k-1}\frac{d_{n,\kappa,\kappa+1}}{V}-\sum_{\kappa=1}^{k-1}D_{n,\kappa}^*$, with $d_{n,\kappa,\kappa+1}$ being the distance between cluster $\xi_{n,\kappa}$ and $\xi_{n,\kappa+1}$. Here, $\frac{d_{n,\kappa,\kappa+1}}{V}$ is the time needed by DBS n to travel in an SLF from cluster $\xi_{n,\kappa}$ to cluster $\xi_{n,\kappa+1}$. $D_{n,\kappa}^*$ is the time needed by DBS n for hovering while serving the users in cluster $\xi_{n,\kappa}$, and this hovering time is given by

$$D_{n,\kappa}^*=\begin{cases}\max\limits_{u\in\mathcal{U}_{n,\kappa}^*}D_{u,n}-\frac{2d_r}{V}, & \text{if }\max\limits_{u\in\mathcal{U}_{n,\kappa}^*}D_{u,n}-\frac{2d_r}{V},\\ 0, & \text{otherwise},\end{cases}\tag{4}$$

[2]The utility at a given DBS is determined by the trajectories of all DBSs since the number of user requests in the course of this DBS n's hovering over a target cluster is determined by the number of active users served by the DBSs previously arrived at this cluster.

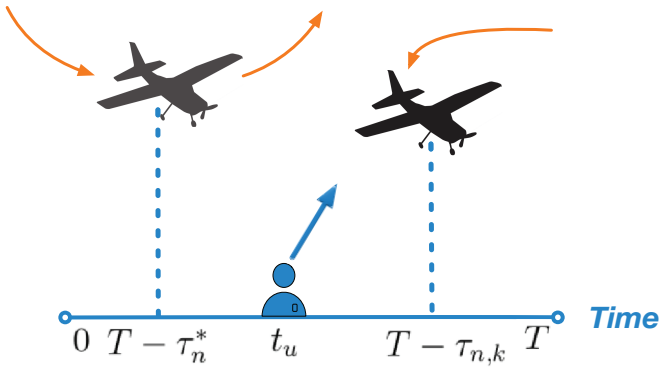


Fig. 2. Snapshot of one user access request. DBSs only connect with active users that have not been served by other DBSs.

where $D_{u,n}=\frac{b_u}{c_{u,n}}$ is the transmission delay of user u, when being served by DBS n. $\max_{u\in\mathcal{U}_{n,\kappa}}D_{u,n}$ is the time that DBS n used to serve all the users in cluster $\xi_{n,\kappa}$. $D_{n,\kappa}^*$ is the time DBS n consumes for SCF hovering, while $\frac{2d_r}{V}$ is the time DBS n consumes its SLF travel within the service area. We define $\mathcal{U}_{n,\kappa}^*=\{u\,|u\in\mathcal{U}_{n,\kappa},T-\tau_n^*\leq t_u\leq T-\tau_{n,\kappa}\}$ as the set of active users that can be served by DBS n in cluster $\xi_{n,\kappa}$.

C. Problem Formulation

In our system model, a group of energy-constrained and myopic DBSs fly independently to cover all the access requests from ground users. The successful service rate achieved by the DBSs is defined as a *team utility*, and is given by

$$G\left(\boldsymbol{\xi}\right)=\sum_{k=1}^{K}\sum_{n\in\mathcal{N}}\mu_{n,k}\left(\boldsymbol{\xi}\right).\tag{5}$$

The goal of the DBSs is to find optimal trajectories that maximize the expected team utility. Next, we first define expected team utility and then we introduce the optimization problem. Let $\pi_n\left(\xi\,|\xi_{n,k},\tau_{n,k}\right)$ be the strategy of DBS n, defined as the probability that DBS n moves toward cluster $\xi\in\mathcal{C}\cup\{O\}$, after successfully serving cluster $\xi_{n,k}$ with available service time $\tau_{n,k}$, and let $\boldsymbol{\pi}=[\pi_n\left(\xi\,|\xi_{n,k},\tau_{n,k}\right)]_{n\in\mathcal{N},k\in\mathcal{K}}$ be the vector of strategies of all DBSs. Then the expected team utility is defined as

$$\overline{G}\left(\boldsymbol{\pi}\right)=\sum_{\boldsymbol{\xi}\in\mathcal{E}}G\left(\boldsymbol{\xi}\right)\prod_{n=1}^{N}\prod_{k=1}^{K}\pi_n\left(\xi\,|\xi_{n,k},\tau_{n,k}\right),\tag{6}$$

where $\mathcal{E}$ is the set of all possible trajectories of the DBSs. As such, the trajectory design problem can be formulated as

$$\max_{\boldsymbol{\pi}}\overline{G}\left(\boldsymbol{\pi}\right),\tag{7}$$

$$\text{s. t.}\sum\nolimits_{\boldsymbol{\xi}\in\mathcal{E}}\prod_{n=1}^{N}\prod_{k=1}^{K}\pi_n\left(\xi\,|\xi_{n,k},\tau_{n,k}\right)=1,\tag{7a}$$

$$\sum\nolimits_{\xi\in\mathcal{C}\cup\{O\}}\pi_n(\xi\,|\xi_{n,k},\tau_{n,k})=1,\forall n\in\mathcal{N},\boldsymbol{\xi}\in\mathcal{E},k\in\mathcal{K},\tag{7b}$$

$$0\leq\pi_n\left(\xi\,|\xi_{n,k},\tau_{n,k}\right)\leq 1,\forall n\in\mathcal{N},\boldsymbol{\xi}\in\mathcal{E},k\in\mathcal{K},\tag{7c}$$

Here, (7a) means that the DBSs must choose trajectories from $\mathcal{E}$. (7b) indicates that, within the considered time duration, each DBS n must choose to serve one cluster in $\mathcal{C}$ or return to the origin. The optimal solution of problem (7) guarantees a maximum expected team utility at the DBSs, and it is called a *team optimal strategy*. Here, we note that the use of traditional optimization algorithms, such as branch and bound or nonlinear programming, is not suitable to solve (7), as problem (7) is non-convex, and the successful service rate $\mu_{n,k}(\boldsymbol{\xi})$ achieved at each DBS is unpredictable with the values of $\boldsymbol{b}$ and $\boldsymbol{t}$ following unknown distributions. Moreover, traditional machine learning algorithms such as Q learning, policy gradient, and echo state networks (ESN) [24]–[26] that have been previously used to solve complex optimization problem are also not suitable to solve the problem (7). This is because those algorithms must be manually adjusted to solve their training tasks, and they cannot find optimal strategies for the DBSs in unseen environments. To solve the non-convex problem (7) formulated in dynamic, unpredictable environments, we propose a distributed meta-trained VD-RL algorithm that finds team optimal strategies by meta training a distributed VD-RL solution equipped with a primary estimation on unseen environments. The distributed VD-RL algorithm updates the DBSs' strategies and estimation on the strategy outcomes at each individual DBS, based on only this DBS's actions and states. The meta training procedure finds an initialization of the policy and value functions for the VD-RL solution. This initialization is close to all optimal policy and value functions at all possible environments and, hence, it enables the VD-RL solution quickly converge in the dynamic environments and reduces the VD-RL solution's cost on time, energy, and the drone hardware. The proposed meta-trained VD-RL algorithm is introduced in the next section.

III. Proposed Value Decomposition-Reinforcement Learning Algorithm with Meta Training

We now introduce a distributed meta-trained VD-RL algorithm, that merges the concept of value decomposition network [27], model agnostic meta-learning [28], with the policy gradient (PG) framework. The traditional PG algorithm can find the optimal trajectory for a single DBS. However, PG cannot find a team optimal strategy for a group of DBSs, as it will direct all the DBSs to one trajectory. In order to design trajectories for multiple DBSs, an algorithm that can reach a team optimal strategy for the DBSs must be proposed. Moreover, to prepare the DBSs for unseen environments, the proposed algorithm must not be overfitted to its training tasks, and it should converge to a team optimal strategy when the team utility function in (5) changes. To address these challenges, we propose a meta-trained VD-RL that coordinates a group of DBSs in various environments. Next, we first introduce the proposed VD-RL algorithm which solves the non-convex problem (7). Then, we explain how to use meta-learning approaches to meta train this VD-RL algorithm in order to allow it to cope with various environments.

A. Value Decomposition based Reinforcement Learning Algorithm

Next, we first introduce the components of the VD-RL algorithm. Then, we explain how the VD-RL algorithm can decompose problem (7) to problems that can be solved at each individual DBS. Finally, we introduce the procedure of using the proposed VD-RL algorithm to solve the non-convex problem in (7).

1) Value decomposition based reinforcement learning components: The proposed VD-RL algorithm consists of seven components

- *Agents*: The agents in VD-RL are the DBSs in set $\mathcal{N}$.
- *States*: Each agent has a state that consists of both its location, represented by the cluster it currently serves i.e. $\xi_{n,k}$, and its energy level, which is captured by the time $\tau_{n,k}$ that this DBS still has in order to return to the origin. Thus, the state of DBS n at step k is given by $\boldsymbol{s}_{n,k} = [\xi_{n,k}, \tau_{n,k}]$. The set of states at all DBSs is $\mathcal{S} = \{\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_K\}$, with $\boldsymbol{S}_k = [\boldsymbol{s}_{1,k}, \boldsymbol{s}_{2,k}, \ldots, \boldsymbol{s}_{N,k}]$ being the matrix of the DBSs' states at step k.
- *Actions*: The action of each agent is the cluster that it seeks to serve, or the origin location that it will return to after serving several clusters. The action chosen by DBS n at step k is given by $a_{n,k} \in \mathcal{C} \cup \{O\}$, while the vector of all DBSs' actions at step k is $\boldsymbol{a}_k = [a_{1,k}, a_{2,k}, \ldots, a_{N,k}]$.
- *Strategy*: The strategy of each DBS is defined as the probability of choosing a given action $a_{n,k} \in \mathcal{C} \cup \{O\}$ at a given state $\boldsymbol{s}_{n,k}$ and is denoted by $\pi_n(a_{n,k} | \boldsymbol{s}_{n,k})$. $\boldsymbol{\pi} = [\pi_n(a_{n,k} | \boldsymbol{s}_{n,k})]_{n \in \mathcal{N}, k \in \mathcal{K}}$ is the vector of strategies of all DBSs.
- *Policy function*: We define a policy function $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}}$ that is a deep neural network parametrized by $\boldsymbol{\theta}_{a,n}$ and is used to generate the strategy of DBS n. This policy function $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}}(a_{n,k} | \boldsymbol{s}_{n,k})$ takes DBS n's state as an input and outputs a strategy for DBS n at this state.
- *Reward*: The reward of each DBS measures the benefit of a selected action. In particular, aiming at maximizing coverage in the considered area, the reward of each DBS is defined as the successful service rate achieved by all the DBSs, which is given by $r(\boldsymbol{a}_k | \mathcal{S}) = \sum_{n \in \mathcal{N}} \mu_{n,k}(\boldsymbol{\xi})$. Here, different from traditional RL algorithms in which each agent only maximizes its own achievable utility, our proposed VD-RL algorithm enables each DBS to maximize the utility of all DBSs, which is also called the team stage reward [3].

[3]Note that, DBS n's successful service rate $\mu_{n,k}$ defined in (3), depends on DBS n's current action $a_{n,k}$, state $\boldsymbol{s}_{n,k}$, and the actions taken by other DBSs before DBS n's arrival at cluster $\xi_{n,k}$. As the DBSs operate asynchronously in the wireless environment as shown in Fig. 3, one DBS may reach step $k' \neq k$ on its trajectory at the time DBS n arrives at its step k. In other words, at time epoch $T - \tau_{n,k}$, the DBSs will reach different steps of their trajectories. The set of DBSs' states at time epoch $T - \tau_{n,k}$ is denoted as $\mathcal{B}_{T-\tau_{n,k}}$. In this case, the team stage reward resulting from action $\boldsymbol{a}_k$ should be $r\left(\boldsymbol{a}_k \middle| \cup_{n \in \mathcal{N}} \mathcal{B}_{T-\tau_{n,k}}\right)$. Hereinafter, we denote it as $r(\boldsymbol{a}_k | \boldsymbol{S})$ for simplicity. It is worthy noting that, within the proposed VD-RL algorithm, the DBSs only record the number of served active users at each step on their trajectories.

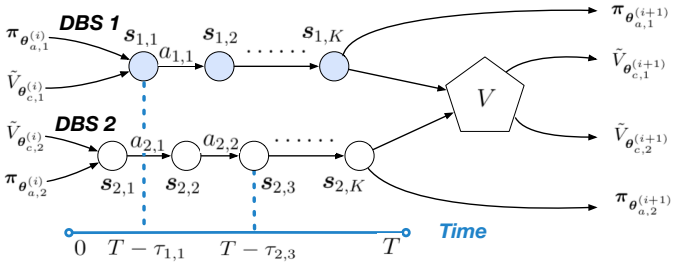


Fig. 3. Illustration of one iteration of the VD-RL algorithm defined in Algorithm 1.

- *Value function*: We define a value function $V(\boldsymbol{S}_k)$ that is a deep neural network used to estimate the DBSs' achievable future rewards at every state $\boldsymbol{S}_k$. In particular, within the proposed VD-RL algorithm, the goal of the DBSs is to find a team optimal strategy that maximizes the expected team utility in (5). To this end, the DBSs must consider the current and also future rewards that can be achieved at every state. Thus, at every state $\boldsymbol{S}_k$, the DBSs seeks to maximize a discounted future reward $\sum_{n\in\mathcal{N}}\sum_{\kappa=k}^{K}\gamma^{k-\kappa+1}r(\boldsymbol{a}_\kappa|\mathcal{S})$, which is estimated by value function $V(\boldsymbol{S}_k)$, with γ being the discounted factor[4].

In VD-RL, the DBSs interact with the unknown wireless environment by selecting actions in $\mathcal{C}\cup\{O\}$ based on the strategy generated by policy function $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}}$. In particular, the DBSs select actions step by step and receive the team stage rewards as feedback. An *experience* defined in vector $\boldsymbol{e}_n = [\boldsymbol{s}_{n,1}, a_{n,1}, r(\boldsymbol{a}_1|\mathcal{S}), \ldots, \boldsymbol{s}_{n,K}, a_{n,K}, r(\boldsymbol{a}_K|\mathcal{S})]$ is collected by each DBS $n\in\mathcal{N}$. The DBSs then update the value function with their collected experiences and update their strategies to maximize future rewards estimated by $V(\boldsymbol{S}_k)$. In particular, the DBSs will update their estimation on future rewards after they return to the origin. At the origin, each DBS n only needs to know the number of active users served by all the DBSs from their last experience to update its estimation on future rewards, it does not have to know the actions taken by other DBSs. However, value function $V(\boldsymbol{S}_k)$ still depends on the states of all DBSs. Hence, each DBS may not be able to train its value function individually. Therefore, we introduce a value decomposition method using which each DBS can train its value function based on its local states.

2) Value decomposition: The VD-RL algorithm needs to find a value function accurately estimate future rewards, and, thus, seeks to minimize the temporal difference (TD) error metric that is defined as follows [29]:

$$A(\boldsymbol{a}_k,\boldsymbol{S}_k) = Nr(\boldsymbol{a}_k|\boldsymbol{S}) + \gamma V(\boldsymbol{S}_{k+1}) - V(\boldsymbol{S}_k) \tag{8}$$

where TD error $A(\boldsymbol{a}_k,\boldsymbol{S}_k)$ is called the *team advantage* of the DBSs at state $\boldsymbol{S}_k$ with action $\boldsymbol{a}_k$. The team advantage measures the difference between the DBSs' future reward estimated by $Nr(\boldsymbol{a}_k|\boldsymbol{S})+\gamma V(\boldsymbol{S}_{k+1})$, and the one estimated by $V(\boldsymbol{S}_k)$. In essence, the team advantage keeps improving the value function $V(\boldsymbol{S}_k)$'s estimation accuracy using new sampled experiences. Moreover, the team advantage also reveals how actions in $\boldsymbol{a}_k$ are better than other action selections by showing the difference between the future reward reached by actions in $\boldsymbol{a}_k$ and the estimated future reward $V(\boldsymbol{S}_k)$. However, to enable the DBSs to update their value functions individually, one must divide the value of function $V(\boldsymbol{S}_k)$ among all DBSs. The value allocated to each DBS must reinforce team beneficial actions, and weaken all other actions at each DBS. Our VD-RL algorithm implements this value decomposition across the DBSs. In particular, the proposed VD-RL algorithm decomposes value $V(\boldsymbol{S}_k)$ under the assumption that this value is the sum of the values attributed to each DBS

$$V(\boldsymbol{S}_k) = \sum_{n\in\mathcal{N}} \tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k}), \tag{9}$$

where $\tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k})$ is the individual value function parametrized by $\boldsymbol{\theta}_{c,n}$ at DBS n. Thus, the team advantage $A(\boldsymbol{a}_k,\boldsymbol{S}_k)$ be expressed as

$$A(\boldsymbol{a}_k,\boldsymbol{S}_k) = Nr(\boldsymbol{a}_k|\boldsymbol{S}) + \gamma\sum_{n\in\mathcal{N}}\tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k+1}) - \sum_{n\in\mathcal{N}}\tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k}). \tag{10}$$

Based on the assumption (9), the DBSs can update their policy and value functions independently.

3) Value decomposition based reinforcement learning solution: When searching for the team optimal strategy that solves problem (7), the VD-RL algorithm needs to find optimal value functions that accurately estimate future rewards at every state, as well as the optimal policy functions that can always yield the actions leading to the highest future rewards at each DBS. In particular, the update of the individual value function $\tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k})$ of each DBS n is given by

$$\begin{aligned}\boldsymbol{\theta}_{c,n}^{(i+1)} &= \boldsymbol{\theta}_{c,n}^{(i)} - \alpha_c^{(i)}\nabla_{\boldsymbol{\theta}_{c,n}^{(i)}}\sum_{k=1}^{K}A^2(\boldsymbol{a}_k,\boldsymbol{S}_k)\\ &= \boldsymbol{\theta}_{c,n}^{(i)} - \alpha_c^{(i)}\nabla_{\boldsymbol{\theta}_{c,n}^{(i)}}\sum_{k=1}^{K}\Bigg(Nr(\boldsymbol{a}_k|\boldsymbol{S}) + \gamma\sum_{n\in\mathcal{N}}\tilde{V}_{\boldsymbol{\theta}_{c,n}^{(i)}}(\boldsymbol{s}_{n,k+1})\\ &\quad - \sum_{n\in\mathcal{N}}\tilde{V}_{\boldsymbol{\theta}_{c,n}^{(i)}}(\boldsymbol{s}_{n,k})\Bigg)^2\\ &= \boldsymbol{\theta}_{c,n}^{(i)} + 2\alpha_c^{(i)}\sum_{k=1}^{K}A(\boldsymbol{a}_k,\boldsymbol{S}_k)\nabla_{\boldsymbol{\theta}_{c,n}^{(i)}}\tilde{V}_{\boldsymbol{\theta}_{c,n}^{(i)}}(\boldsymbol{s}_{n,k}),\end{aligned}$$

where $\boldsymbol{\theta}_{c,n}^{(i)}$ is the value function parameter of DBS n and $\alpha_c^{(i)}$ is the value function parameter update step size, at the

[4]The discounting factor γ determines the scale of these steps. In particular, a discount factor close to 1 provides a long-term goal that accumulates rewards far into the future, while a discount factor close to 0 provides a short-sighted goal with which the DBSs only focus on immediate rewards. In fact, a proper scaling/discounting of future rewards can help a learning algorithm converge faster to the optimal solutions. If one uses the exact future rewards, the DBSs will only focus on their long-term goal and, thus, they would directly head to the cluster with most active users without stopping at other clusters. This would lead to a lower overall successful service rate (which can be seen, for example, in the numerical results in our conference paper [1]).

Algorithm 1 VD-RL algorithm for trajectory design at one realization of user access request.

Input: User locations, time constraints.

Init: Initialize value functions $\tilde{V}_{\boldsymbol{\theta}_{c,n}^{(1)}}$, and policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}^{(1)}}$, for $n \in \mathcal{N}$.

1: **for** VD-RL training epoch $i = 1 : I$ **do**
2: The DBSs carry out an experience by generating a sequence of actions that are randomly selected based on current policy functions.
3: Calculate advantage A based on (8).
4: **for** Each DBS $n = 1 : N$ **do**
5: Update individual value function with (III-A3).
6: Calculate individual advantages $\tilde{A}_n\left(a_{n,k}, \boldsymbol{s}_{n,k}\right)$.
7: Update policy function with (11).
8: **end for**
9: **end for**

i-th VD-RL training iteration [5].

After precisely estimating the future team stage rewards using $V\left(\boldsymbol{S}_k\right)$, and attributing the value to each one of the DBS, the policy function of each DBS can be updated. Let $\tilde{A}_n\left(a_{n,k}, \boldsymbol{s}_{n,k}\right) = r\left(\boldsymbol{a}_k \left| \boldsymbol{S}\right.\right) + \gamma \tilde{V}_{\boldsymbol{\theta}_{c,n}}\left(\boldsymbol{s}_{n,k+1}\right) - \tilde{V}_{\boldsymbol{\theta}_{c,n}}\left(\boldsymbol{s}_{n,k}\right)$ be the individual advantage at BS n at state $\boldsymbol{s}_{n,k}$ with action $a_{n,k} \in \boldsymbol{a}_k$. Hence, we have $A\left(\boldsymbol{a}_k, \boldsymbol{S}_k\right) = \sum_{n\in\mathcal{N}} \tilde{A}_n\left(a_{n,k}, \boldsymbol{s}_{n,k}\right)$. Based on the the policy gradient theorem [26], the update on DBS n's policy function parameters is given by

$$\boldsymbol{\theta}_{a,n}^{(i+1)} = \boldsymbol{\theta}_{a,n}^{(i)} + \alpha_a^{(i)} \sum_{k=1}^{K} \tilde{A}_n\left(a_{n,k}, \boldsymbol{s}_{n,k}\right) \nabla_{\boldsymbol{\theta}_{a,n}^{(i)}} \log \pi_{\boldsymbol{\theta}_{a,n}^{(i)}}\left(a_{n,k} \left| \boldsymbol{s}_{n,k}\right.\right), \tag{11}$$

where $\boldsymbol{\theta}_{a,n}^{(i)}$ is the policy function parameter at DBS n and $\alpha_a^{(i)}$ is the policy function parameter update step size, at the i-th VD-RL training iteration.

The proposed VD-RL solution is summarized in Algorithm 1. At the beginning of the algorithm, each DBS serves its users with a randomly initialized value function $V_{\boldsymbol{\theta}_{c,n}^{(1)}}$ and policy function $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}^{(1)}}$. The successful service rates that the DBSs achieved with the selected policy functions are calculated and recorded by the DBSs upon their return to the origin. The DBSs update their value and policy function parameters based on (III-A3) and (11), using the recorded successful service rates. Then, the DBSs serve the users with the updated policy functions. Note that the proposed VD-RL algorithm deploys a mini-batch training mechanism. In particular, using the proposed algorithm, the DBS updates its policy over a whole experience in order to reduce the variance caused by the action sampling, as well as the need to store a big dataset.

[5]Note that, here, $Nr\left(\boldsymbol{a}_k \left| \boldsymbol{S}\right.\right) + \gamma \sum_{n\in\mathcal{N}} \tilde{V}_{\boldsymbol{\theta}_{c,n}}\left(\boldsymbol{s}_{n,k+1}\right)$ acts as the supervised terms (targets) in the training procedure of the value functions, and, thus, do not take derivations. Also, the limited situational awareness only allows the DBSs to define their individual value functions based on their local states (otherwise, significant coordination and overhead would be needed which is impractical for a drone network). The proposed mathematical induction and mini-batch training procedure allows the DBSs update their individual value functions based on their local states.

Thus, while flying over the target area, the DBSs will, step by step, select clusters to serve by sampling actions from their policy functions. Upon their return to the origin, the DBSs will exchange their achieved rewards, and update their policy and value functions by accumulating these rewards. Also, for a given DBS, carrying out an experience means that this DBS will serve one realization of user requests, and then, record its actions, states, and achieved rewards from this service. The recorded actions, states, and rewards constitute the experience of a given DBS.

4) Convergence and complexity analysis: Next, we show that the proposed VD-RL algorithm is guaranteed to converge to a local team optimal strategy.

Proposition 1. The proposed distributed VD-RL algorithm is guaranteed to converge to a local optimal solution of problem (7), if the following conditions are satisfied:

1) Value function $\tilde{V}_{\boldsymbol{\theta}_{c,n}}\left(\boldsymbol{s}_{n,k}\right)$ converges to a local minimum with $\nabla_{\boldsymbol{\theta}_{c,n}} A^2\left(\boldsymbol{a}_k, \boldsymbol{S}_k\right) = 0$, and advantage function $A\left(\boldsymbol{a}_k, \boldsymbol{S}_k\right) \neq 0$.
2) $\max\limits_{a_{n,k}\in\mathcal{C}, \boldsymbol{s}_{n,k}\in\mathcal{S}} \left| \frac{\partial \prod\limits_{n\in\mathcal{N}} \pi_{\boldsymbol{\theta}_{a,n}}\left(a_{n,k}\left|\boldsymbol{s}_{n,k}\right.\right)}{\partial\theta_i \partial\theta_j'} \right| < \infty$, for any elements θ_i and θ_j in the policy function parameter vector $\boldsymbol{\theta}_a = [\boldsymbol{\theta}_{a,1}, \boldsymbol{\theta}_{a,2}, \ldots, \boldsymbol{\theta}_{a,N}]$.
3) $\lim_{i\to\infty} \alpha_c^{(i)} = 0$, $\sum_{i=1}^{\infty} \alpha_c^{(i)} = \infty$, $\lim_{i\to\infty} \alpha_a^{(i)} = 0$ and $\sum_{i=1}^{\infty} \alpha_a^{(i)} = \infty$.

Proof. See Appendix A. □

From Proposition 1, we can see that the proposed VD-RL algorithm is guaranteed to converge to a local optimal solution of problem (7), by decomposing value $V\left(\boldsymbol{S}_k\right)$ to each DBSs.

With regards to the complexity of the proposed VD-RL algorithm, one can see that only the successful service rate and estimated future reward, i.e. individual value, reached by each DBS at each step on their trajectories will be shared and transmitted among the DBSs. Thus, the VD-RL algorithm reduces the dimensionality of the problem by updating the policy and value functions at each DBS based only on this DBS's actions and states. Note that, the studied multi-agent problem is much more complex than a single agent problem, as the exponential growth on the action and state spaces in multi-agent problem causes a curse of dimensionality[6]. Nonetheless, our approach allows to reduce this multi-agent dimensionality to that of a single-agent problem. In essence, the complexity of the VD-RL solution is $\mathcal{O}\left(\upsilon\left(n_c + n_a\right) C\right)$, where υ is the iteration when the proposed VD-RL algorithm converged, while n_c and n_a are the number of elements in $\boldsymbol{\theta}_{c,n}$ and $\boldsymbol{\theta}_{a,n}$. C is the time complexity of calculating the gradient of each element in $\boldsymbol{\theta}_{c,n}$ and $\boldsymbol{\theta}_{a,n}$. This complexity is considerably low since, as already mentioned, it is similar to that of a single agent policy gradient algorithm.

[6]Curse of dimensionality means that the neural networks' estimation error increases when they deal with high dimensional data [30]

In summary, using the VD-RL algorithm, the DBSs update their own policy and value functions locally and can finally reach a local optimal solution of problem (7). However, when the environment changes, the local optimal strategy reached by the VD-RL algorithm will no longer be the strategy that maximizes the expected coverage. In this case, the whole procedure in Algorithm 1 must be performed again to solve problem (7). As the studied wireless environment is highly dynamic, the VD-RL algorithm in Algorithm 1 must be repeatedly performed to search a team optimal strategy in every environment. This increases the complexity of finding a VD-RL solution to a time complexity $\mathcal{O}\left(\upsilon\left(n_c+n_a\right)CX\right)$, when the user access requests change X times. In order to reduce this time complexity, we use the framework of meta-learning [28]. In particular, we propose a meta training procedure that can supplement the proposed algorithm by finding an initialization of the policy and value functions for the VD-RL solution. The meta-trained initialization are close to optimal policy and value functions at all possible environments which enables the VD-RL algorithm to quickly converge in dynamic environments.

B. Meta Training Procedure

Next, we introduce our meta-learning approach that meta trains a VD-RL solution using model agnostic meta-learning (MAML) [28]. The proposed meta training method seeks a VD-RL solution capable of quickly reaching team optimal strategies in various environments. This meta training procedure prepares the VD-RL solution with a set of well established initial policy and value functions with suitable estimation on a distribution of user request realizations, i.e. $p\left(\mathcal{Z}\right)$. These initial policy and value functions are close to the optimal ones that provide team optimal strategies to each of the user request realizations in $\mathcal{Z}$. During the meta training procedure, a realization $\boldsymbol{z}_j$ is first sampled from $p\left(\mathcal{Z}\right)$. Then, the DBSs collect experiences $\boldsymbol{e}_{n,j}$ using their initial policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}^{(1)}}$, with the consideration that ground users are requesting access based on $\boldsymbol{z}_j$. The policy and value functions at the DBSs are updated using Algorithm 1, based on the collected experiences $\boldsymbol{e}_{n,j}$. The updated policy and value functions at DBS n within $\boldsymbol{z}_j$ are denoted as $\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}}$, and $\boldsymbol{\pi}_{\boldsymbol{\theta}'_{a,n,j}}$, respectively. The updated policy and value functions are then tested on new experiences $\boldsymbol{e}'_{n,j}$ with actions sampled using $\boldsymbol{\pi}_{\boldsymbol{\theta}'_{a,n,j}}$, at current realization $\boldsymbol{z}_j$. The feedback from such tests at DBS n will be the values of loss functions that measure the distance from the updated policy and value functions to the optimal policy and value functions that produce team optimal strategies. These are given by

$$\tilde{L}_{c,n}\left(\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}},\boldsymbol{z}_j\right)=\sum_{k=1}^{K}\left(r\left(\boldsymbol{a}_k^{\left(\boldsymbol{e}'_{n,j}\right)}\middle|\boldsymbol{S}_k^{\left(\boldsymbol{e}'_{n,j}\right)}\right)+\gamma\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}}\left(\boldsymbol{s}_{n,k+1}^{\left(\boldsymbol{e}'_{n,j}\right)}\right)-\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}}\left(\boldsymbol{s}_{n,k}^{\left(\boldsymbol{e}'_{n,j}\right)}\right)\right)^2, \tag{12}$$

Algorithm 2 Meta training procedure for the trajectory design problem.

Input: A distribution of user requests $p\left(\mathcal{Z}\right)$, user locations, time constraints.

Init: Initialize value functions $\tilde{V}_{\boldsymbol{\theta}_{c,n}^{(1)}}$, and policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}^{(1)}}$, for $n\in\mathcal{N}$.

1: **for** Meta training epoch $i=1:I$ **do**
2: **for** Environment sampling epoch $j=1:J$ **do**
3: Sample an user request realization $\boldsymbol{z}_j\sim\mathcal{Z}$.
4: Carry out an experience by generating a sequence of actions that are randomly selected based on current policy functions, i.e. $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}^{(i)}}$, for $n\in\mathcal{N}$.
5: Calculate advantage A based on (8).
6: **for** Each DBS $n=1:N$ **do**
7: Update individual value functions

$$\boldsymbol{\theta}_{c,n,j}^{(i)'}=\boldsymbol{\theta}_{c,n}^{(i)}+\alpha_c\nabla_{\boldsymbol{\theta}_{c,n}^{(i)}}\sum_{k=1}^{K}A^2\left(\boldsymbol{a}_k^{\left(\boldsymbol{e}_{n,j}\right)},\boldsymbol{S}_k^{\left(\boldsymbol{e}_{n,j}\right)}\right),$$

and policy functions

$$\boldsymbol{\theta}_{a,n,j}^{(i)'}=\boldsymbol{\theta}_{a,n}^{(i)}+\alpha_a\sum_{k=1}^{K}\tilde{A}_n\left(a_{n,k}^{\left(\boldsymbol{e}_{n,j}\right)},\boldsymbol{s}_{n,k}^{\left(\boldsymbol{e}_{n,j}\right)}\right)\nabla_{\boldsymbol{\theta}_{a,n}^{(i)}}\log\pi_{\boldsymbol{\theta}_{a,n}^{(i)}}\left(a_{n,k}^{\left(\boldsymbol{e}_{n,j}\right)}\middle|\boldsymbol{s}_{n,k}^{\left(\boldsymbol{e}_{n,j}\right)}\right).$$

8: Carry out an experience by generating a sequence of actions that are randomly selected based on updated policy functions, i.e. $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n,j}^{(i)'}}$.
9: Calculate loss $\tilde{L}_{c,n}\left(\tilde{V}_{\boldsymbol{\theta}_{c,n,j}^{(i)'}},\boldsymbol{z}_j\right)$ and $\tilde{L}_{a,n}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n,j}^{(i)'}},\boldsymbol{z}_j\right)$ with (12) and (13).
10: **end for**
11: **end for**
12: **for** Each DBS $n=1:N$ **do**
13: Update initial value parameters $\boldsymbol{\theta}_{c,n}^{(i+1)}=\boldsymbol{\theta}_{c,n}^{(i)}-\beta\nabla_{\boldsymbol{\theta}_{c,n}^{(i)}}\sum_{j=1}^{J}\tilde{L}_{c,n}\left(\tilde{V}_{\boldsymbol{\theta}_{c,n,j}^{(i)'}},\boldsymbol{z}_j\right)$, and policy parameters $\boldsymbol{\theta}_{a,n}^{(i+1)}=\boldsymbol{\theta}_{a,n}^{(i)}-\beta\nabla_{\boldsymbol{\theta}_{a,n}^{(i)}}\sum_{j=1}^{J}\tilde{L}_{a,n}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n,j}^{(i)'}},\boldsymbol{z}_j\right)$.
14: **end for**
15: **end for**
16: **return** Optimal initial value functions $\tilde{V}_{\boldsymbol{\theta}_{c,n}^{*}}$, and initial policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}_{a,n}^{*}}$, for $n\in\mathcal{N}$.

$$\tilde{L}_{a,n}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}'_{a,n,j}},\boldsymbol{z}_j\right)=\sum_{k=1}^{K}\left(r\left(\boldsymbol{a}_k^{\left(\boldsymbol{e}'_{n,j}\right)}\middle|\boldsymbol{S}_k^{\left(\boldsymbol{e}'_{n,j}\right)}\right)+\gamma\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}}\left(\boldsymbol{s}_{n,k+1}^{\left(\boldsymbol{e}'_{n,j}\right)}\right)-\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}}\left(\boldsymbol{s}_{n,k}^{\left(\boldsymbol{e}'_{n,j}\right)}\right)\right)\log\pi_{\boldsymbol{\theta}'_{a,n,j}}\left(a_{n,k}^{\left(\boldsymbol{e}'_{n,j}\right)}\middle|\boldsymbol{s}_{n,k}^{\left(\boldsymbol{e}'_{n,j}\right)}\right), \tag{13}$$

where $\boldsymbol{a}_k^{\left(\boldsymbol{e}'_{n,j}\right)}$ and $\boldsymbol{S}_k^{\left(\boldsymbol{e}'_{n,j}\right)}$ are, respectively, the k-th action and state of DBS n within experience $\boldsymbol{e}'_{n,j}$. By minimizing the loss functions in (12) and (13), the proposed meta training method updates the policy and value function parameters toward the parameters of optimal policy and value functions in various user access request realizations sampled from $p\left(\mathcal{Z}\right)$. More concretely, the objective of the meta training procedure is

$$\min_{\boldsymbol{\theta}_c,\boldsymbol{\theta}_a}\sum_{\boldsymbol{z}_j\sim p(\mathcal{Z})}\sum_{n=1}^{N}\tilde{L}_{c,n}\left(\tilde{V}_{\boldsymbol{\theta}'_{c,n,j}},\boldsymbol{z}_j\right)+\tilde{L}_{a,n}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}'_{a,n,j}},\boldsymbol{z}_j\right), \tag{14}$$

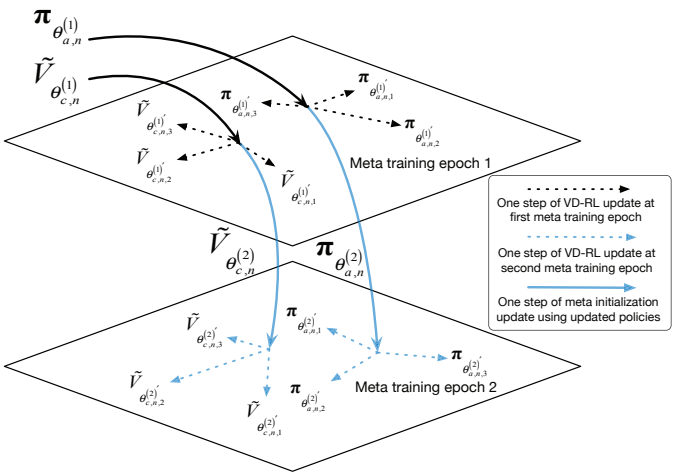


Fig. 4. Illustration of the proposed meta-training method at DBS n with $J = 3$. Here, 3 different user requests are drawn from $p(\mathcal{Z})$ at each training episode. $\tilde{V}_{\boldsymbol{\theta}^{(1)\prime}_{c,n,1}}$ and $\boldsymbol{\pi}_{\boldsymbol{\theta}^{(1)\prime}_{a,n,1}}$ are the updated policy and value functions in the first sampled environment,

where $\boldsymbol{\theta}_c = [\boldsymbol{\theta}_{c,1}, \boldsymbol{\theta}_{c,2}, \ldots, \boldsymbol{\theta}_{c,N}]$ is the vector of value function parameters at all DBSs. $\boldsymbol{\theta}'_{c,n,j} = \boldsymbol{\theta}_{c,n} - \alpha_c \nabla_{\boldsymbol{\theta}_{c,n}} \sum_{k=1}^{K} A^2\left(\boldsymbol{a}_k^{(\boldsymbol{e}_{n,j})}, \boldsymbol{S}_k^{(\boldsymbol{e}_{n,j})}\right)$ is the updated value function parameters of DBS n, based on experience $\boldsymbol{e}_{n,j}$. $\boldsymbol{\theta}'_{a,n,j} = \theta_{a,n} + \alpha_a \sum_{k=1}^{K} \tilde{A}_n\left(a_{n,k}^{(\boldsymbol{e}_{n,j})}, \boldsymbol{s}_{n,k}^{(\boldsymbol{e}_{n,j})}\right) \nabla_{\theta_{a,n}} \log \pi_{\theta_{a,n}}\left(a_{n,k}^{(\boldsymbol{e}_{n,j})} \middle| \boldsymbol{s}_{n,k}^{(\boldsymbol{e}_{n,j})}\right)$ is the updated policy function parameter of DBS n, based on experience $\boldsymbol{e}_{n,j}$. Here, we can see that the optimization variables in (14) are initial policy and value function parameters $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_a$. That is, the meta-learning approach seeks to find the optimal initial function parameters of the VD-RL algorithm so as to minimize the estimation errors captured in (12) and (13), for different environments that can be sampled from $p(\mathcal{Z})$. This parameter initialization, i.e. the meta-trained policy and value functions, will make it easier for the VD-RL solution to find the team optimal strategies in unseen environments.

To solve problem (14), a standard stochastic gradient descent method is applied. Fig. 4 shows one iteration for solving problem (14). For this example, at each meta training iteration, the meta training method performs one step stochastic gradient descent on (14) over the initial policy and value function parameters $\boldsymbol{\theta}_{c,n}$ and $\boldsymbol{\theta}_{a,n}$ based on sampled experiences in J different environments. The full procedure of using meta training method with VD-RL is summarized in Algorithm 2. [7] In this meta training procedure, the DBSs collect their experiences from serving J different user requests drawn from $p(\mathcal{Z})$, with initial value functions $\tilde{V}_{\boldsymbol{\theta}^{(1)}_{c,n}}$ and policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}^{(1)}_{a,n}}$. After serving one sampled user request j, the DBSs execute one step VD-RL update and get new value functions $\tilde{V}_{\boldsymbol{\theta}^{(1)\prime}_{c,n,j}}$ and policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}^{(1)\prime}_{a,n,j}}$. The updated policy and value functions are evaluated by the loss functions defined in (12) and (13), based on experiences collected with strategy $\boldsymbol{\pi}_{\boldsymbol{\theta}^{(1)\prime}_{a,n,j}}$. The meta training procedure in Algorithm 2 seeks to finding a set of initial policy and value functions that are close to the optimal policy and value functions at every user request realization. Equipped with this initialization, the VD-RL can now find a local optimal solution of problem (7) in unseen environment using a small number of update steps in Algorithm 1. Thus, the meta-training method minimizes the losses that are collected from the sampled user request realizations, by updating the value and policy parameters, i.e. $\boldsymbol{\theta}^{(1)}_{c,n}$, $\boldsymbol{\theta}^{(1)}_{a,n}$, in the opposite direction of the gradient of the losses. After one step of this update, the value and policy parameters, i.e. $\boldsymbol{\theta}^{(2)}_{c,n}$, $\boldsymbol{\theta}^{(2)}_{a,n}$, will be closer to the optimal policy and value functions in the sampled environment. Subsequently, the DBSs will start serving user requests with value functions $\tilde{V}_{\boldsymbol{\theta}^{(2)}_{c,n}}$, and policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}^{(2)}_{a,n}}$. Then, the DBSs collect their experience from serving different user requests, based on which they can keep updating the policy and value functions, as shown in Fig. 4, until convergence[8]. Using our proposed meta training procedure, the DBSs finally find a VD-RL solution with optimal initializations of the policy and value functions, i.e. $\tilde{V}_{\boldsymbol{\theta}^{*}_{c,n}}$, and $\boldsymbol{\pi}_{\boldsymbol{\theta}^{*}_{a,n}}$. The complexity of the meta training procedure is $\mathcal{O}\left(2\upsilon_m\left(n_c + n_a\right)CJ\right)$, with υ_m being the number of iterations that the meta training method needs for convergence. This complexity is considerably low, as the proposed meta training mechanism does not needs any additional neural networks in the training procedure. Also, since the DBSs can collect experiences on serving ground user' daily requests, the proposed meta training procedure's complexity will not introduce additional costs in terms of time, energy, or DBS hardware.

IV. Simulation Results and Analysis

For our simulations, we consider a scenario with five DBSs serving $U = 300$ mobile users. The main simulation parameters are listed in Table I. In particular, we assume the quantity of each user request follows uniform distribution over the interval $[20, 600]$ Mbits, the occurrence of each user request follows Gaussian distribution with a standard deviation 1, and a mean ranges from 0 to 40 minutes. At every independent run of the Monte Carlo experiment studied in this section, we repeatedly deploy the users at some random locations that are uniformly distributed in every cluster, and sample a user access request realization from the assumed distribution. The value and policy functions at each DBSs are formulated by feed forward neural networks. The results of the proposed VD-RL algorithm are compared with the independent actor critic algorithm (IAC) [17] and the monotonic value function

[7]Note that, as the meta training procedure seeks to learn a user request distribution $p(\mathcal{Z})$ in Algorithm 2, all the user requests are sampled from the same distribution $p(\mathcal{Z})$.

[8]Here, we need to point out that, a machine learning model reaches convergence when its training loss falls within an error range around the final value. Thus, we consider that the meta training model has converged when additional training will not improve the model, at which point the meta training procedure is completed.

TABLE I
SIMULATION PARAMETERS [23]

Parameter	Value	Parameter	Value
P_u	20 dBm	V_s	30 m/s
N_0	-170 dBm/Hz	B	1 MHz
μ_{LoS},	1.6	δ_{LoS}	8.41
μ_{NLoS},	23	δ_{NLoS}	33.78

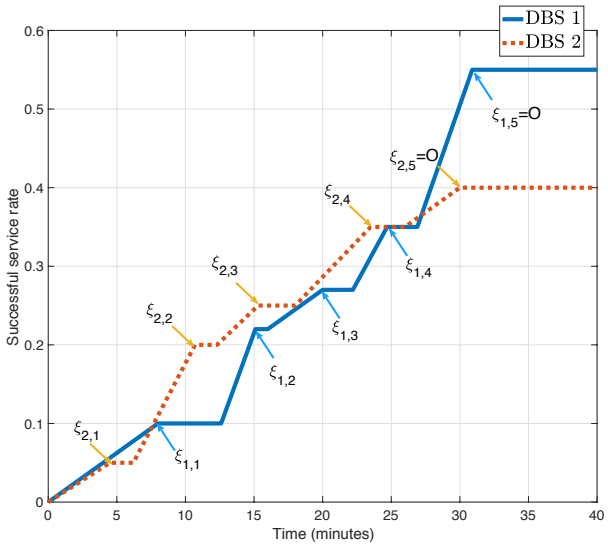
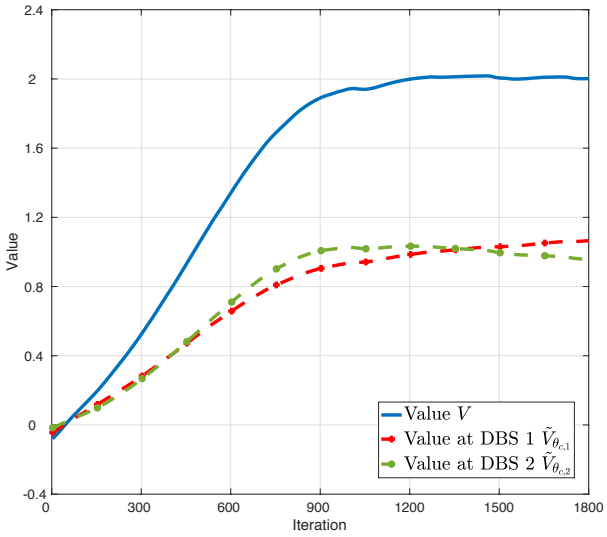


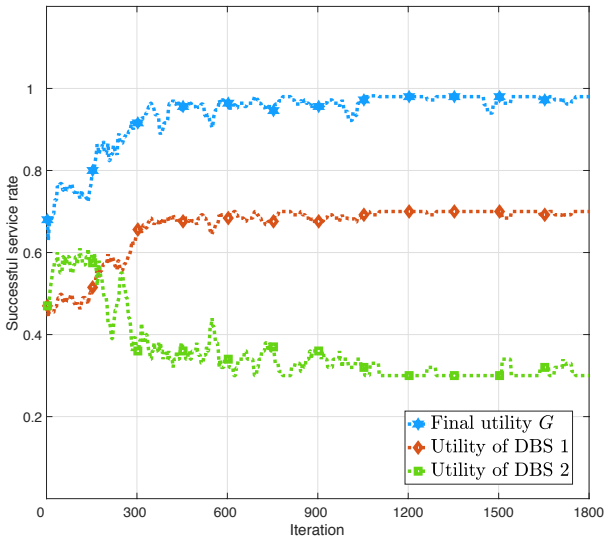
Fig. 5. Successful service rate achieved at each DBS over time, i.e. the time DBSs consume on flying over the simulated area.

factorisation algorithm, called Q mix [18]. The IAC algorithm updates one policy function and one value function for all of the agents. It allows each DBS updates their policies relying only on their local actions and states, but toward the DBSs' maximal individual benefits. The comparison between the proposed algorithm and the IAC algorithm shows the gains one can get from the value decomposition scheme. The Q mix algorithm uses the same value and policy update procedure as our proposed VD-RL algorithm, but estimates the summation in (9) using a neural network. The results of the proposed meta-trained VD-RL are further compared to the results from the pre-trained VD-RL algorithm [31], the original VD-RL algorithm, as well as oracle results. Within the pre-trained VD-RL algorithm, the VD-RL algorithm starts from the optimal policy and value functions at former tasks. Here, the pre-trained VD-RL algorithm trains a VD-RL within exactly the same environments sampled at our meta training procedure. For the original VD-RL algorithm, the DBSs start with randomly initialized policy and value functions. The oracle results present the DBSs' performance with optimal strategies in the considered environment. All statistical results are averaged over a large number of independent runs.

Fig. 5 shows how successful service rates increase with time, in a network with two DBSs. From Fig. 5, we can see that the successful service rate achieved by each DBS increases when the DBS arrives at a new cluster $\xi_{n,k}$. Fig. 5 also shows that the two DBSs spend different amount of time on flying across clusters and serving clusters. In other words, the DBSs operate asynchronously in the considered area. In particular,



(a)

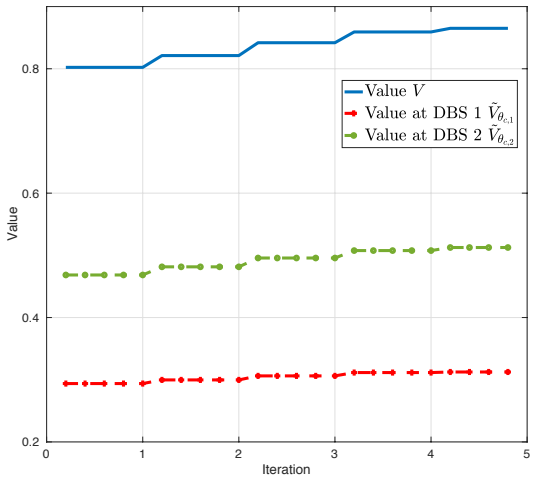


(b)

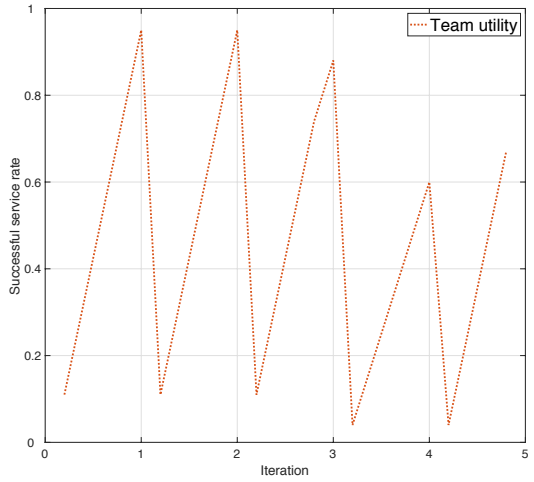
Fig. 6. The value decomposition learned by the proposed VD-RL algorithm. (a) Estimated future rewards decomposed to each DBS, and (b) utility achieved by each DBS, at each iteration of VD-RL.

when DBS 2 starts to serve its second cluster, DBS 1 is still serving its first cluster.

Fig. 6 shows how the decomposition of the value function $V(\boldsymbol{S}_k)$ is learned using the proposed VD-RL algorithm. Here, the value of function $V(\boldsymbol{S}_0)$ is the estimation of the discounted future reward at the initial state $\boldsymbol{S}_0$. In Fig. 6, the value of function $V(\boldsymbol{S}_0)$, as well as the individual values of functions $\tilde{V}_{\boldsymbol{\theta}_{c,1}}(\boldsymbol{S}_0)$ and $\tilde{V}_{\boldsymbol{\theta}_{c,2}}(\boldsymbol{S}_0)$ follow the same trend of as the team utility of all DBSs. This implies that the individual value functions also estimate the team utility. With such values, the DBSs can independently find the strategies that maximize the team utility without sharing information such as actions and states. In particular, at the convergence of the VD-RL algorithm, DBS 2 sacrifices its own utility for team optimality. From Fig. 6, we can also observe that, with the VD-RL algorithm, the individual value at DBS 1 keeps



(a)



(b)

Fig. 7. Value decomposition within 5 VD-RL iteration. (a) Estimated future rewards decomposed to each DBS, (b) utility achieved by each DBS, at each iteration of VD-RL.

increasing, yet the successful service rate at DBS 2 increases at first and then decreases while reaching convergence. The sum of individual values at two DBSs always equals to the value of $V(\boldsymbol{S}_k)$, which follows the trend of the team utility. This is due to the fact that, in the proposed VD-RL algorithm, each DBS is rewarded with the utility achieved by all the DBSs, i.e. team utility. Thus, the DBSs are training their own policies toward a maximal team utility instead of their own utilities. In this case, DBS 2 will be prone to choose policies that yield a lower individual utility but a higher team utility. In other words, the VD-RL algorithm learns to decompose the value function into two individual values, each of which can direct each DBS to independently finds its strategy that maximizes the team utility, instead of its individual utility.

Fig. 7 shows the value decomposition within 5 VD-RL training iterations. In this figure, we can see that the value of $V(\boldsymbol{S}_0)$ increases at the first step within each VD-RL training iteration. This is because, function $V(\boldsymbol{S}_0)$ estimates the cumulative future team reward that the DBSs can get

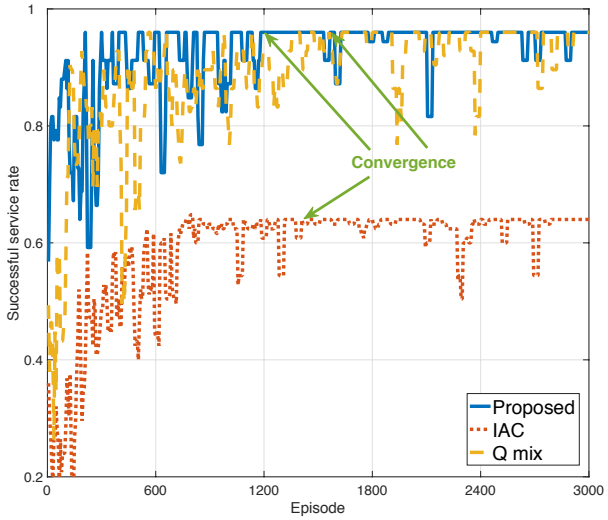


Fig. 8. Convergence of all considered multi-agent RL algorithms. This figure clearly shows that the proposed algorithm converges at faster speed compared to IAC and Q mix.

at the initial state. Thus, the DBSs update value function $V(\boldsymbol{S}_0)$ using the rewards that they achieved from serving the ground users at the last training iteration, once they return to the origin. Fig. 7 also shows that the decomposed individual value $\tilde{V}_{\boldsymbol{\theta}_{c,2}}(\boldsymbol{S}_0)$ at DBS 2 is much larger than the individual value $\tilde{V}_{\boldsymbol{\theta}_{c,1}}(\boldsymbol{S}_0)$ at DBS 1. This is because DBS 2 selects the actions that are beneficial for the team utility. A large individual value $\tilde{V}_{\boldsymbol{\theta}_{c,2}}(\boldsymbol{S}_0)$ reinforces such actions to maintain a high team utility. However, DBS 1 does not select actions that improve the team utility. Hence, the individual value $\tilde{V}_{\boldsymbol{\theta}_{c,1}}(\boldsymbol{S}_0)$ attributed to DBS 1 is small, which keeps DBS 1's strategy unchanged. DBS 1 will, thus, act "greedily" by selecting all possible actions with its current strategy, until it finds the optimal one.

In Fig. 8, we show the convergence of the proposed VD-RL algorithm. In this figure, we can see that the proposed VD-RL algorithm requires approximately $1,300$ iterations to reach convergence, which improves the convergence speed by up to 30.6% compared to the Q mix algorithm. This stems from the fact that the neural network used to estimate the summation in (9) remarkably increases the complexity of Q mix. Meanwhile, Fig. 8 also shows that the proposed VD-RL algorithm yields a 53.2% higher final utility than IAC. This is because the VD-RL algorithm can find a team optimal strategy to maximize the team reward. The IAC algorithm, however, find a strategy that maximize the DBSs' individual utilities. From Fig. 8, we also observe that the proposed VD-RL algorithm has a similar convergence speed to the IAC algorithm. This stems from the fact that, in the VD-RL and IAC algorithm, each DBS updates the policy and value functions using its own experience, and, thus, the proposed VD-RL algorithm and the IAC algorithm all have low time complexities that are comparable to a single agent RL algorithm.

Fig. 9 shows a snapshot of the trajectories resulting from the proposed meta-trained VD-RL, and the pre-trained VD-

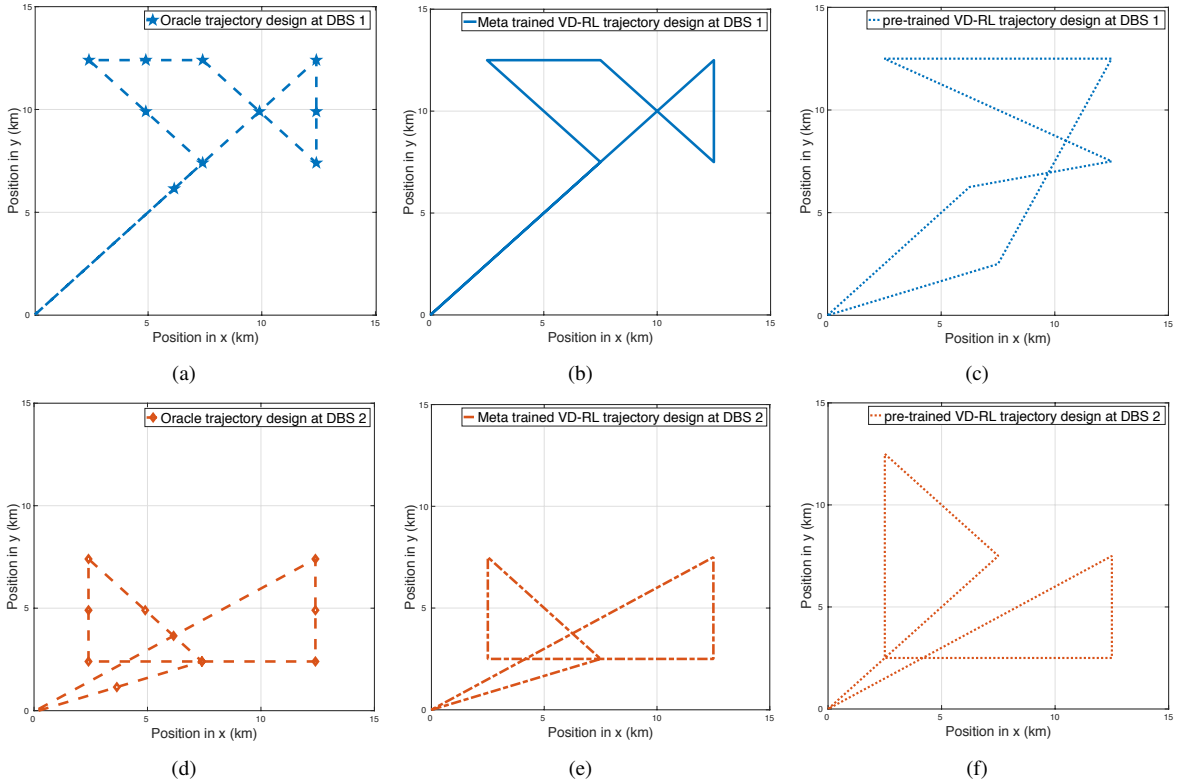


Fig. 9. Snapshots of oracle trajectories (shown in (a) (d)), the trajectories designed by meta-trained (shown in (b) (e)), and pre-trained (shown in (c) (f)) VD-RL, with (a)-(c) being trajectory designs at DBS 1, (d)-(e) being the ones at DBS 2.

RL algorithm in an environment that was not experienced during the meta training procedure. Here, the trajectories in Figs. 9(b), 9(c), 9(e), and 9(f) are selected, respectively, at the 100-th iteration of both considered algorithms. In this figure, we can see that, when faced with an unseen environment, the meta-trained VD-RL scheme can effectively find the optimal trajectories for the DBSs within a considerably small number of iterations. However, the trajectories resulting from the pre-trained VD-RL are still far from the optimal trajectories. Fig. 9 also shows that the proposed meta-training method can adapt the DBSs in unseen environments much faster than the pre-trained VD-RL algorithm. This is because the meta-training method can find a policy and value function parameter initialization that is close to optimal policy and value functions at all possible user requests in $p(\mathcal{Z})$. By using this meta-learning-based initialization, the meta-trained VD-RL can reach the optimal policy, value functions and find a team optimal strategy in an unseen environment within $p(\mathcal{Z})$, using a small number of iterations.

In Fig. 10, we show the convergence of the meta-trained VD-RL. From this figure, we can see that, with initial value functions $\tilde{V}_{\boldsymbol{\theta}^*_{c,n}}$ and policy functions $\boldsymbol{\pi}_{\boldsymbol{\theta}^*_{a,n}}$ provided by the meta training procedure in Algorithm 2, the meta-trained VD-RL converges at approximately the 700-th iteration, which is 36.4%, and 53.8% faster than the pre-trained VD-RL algorithm and the original VD-RL algorithm. Fig. 10 also shows that the meta-trained VD-RL achieves a successful service rate that is equal to the one reached by the original

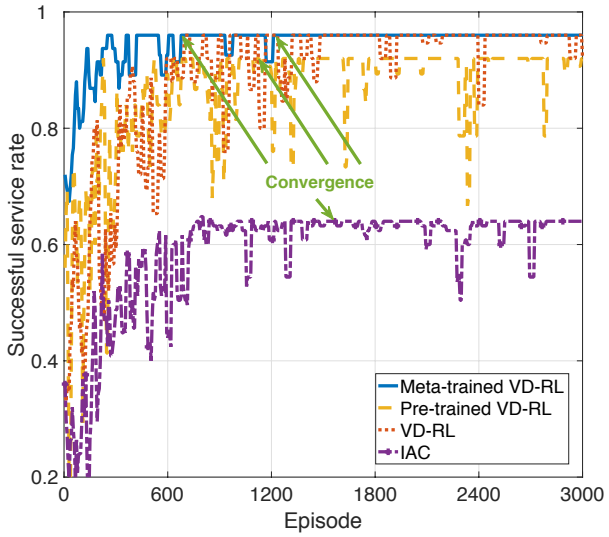


Fig. 10. Convergence of the meta-trained VD-RL, compared to pre-trained VD-RL, VD-RL and IAC. This figure clearly shows that the proposed meta training mechanism significantly improves the convergence speed of VD-RL.

VD-RL algorithm. The successful service rate achieved by the meta-training method is 9.2% and 53.2% higher than the one achieved by, respectively the pre-trained VD-RL algorithm and the IAC algorithm. This gain stems from the fact that the meta-training method trains a set of policy and value functions with proper estimation on various user access requests. By starting the VD-RL policy updating procedure from such

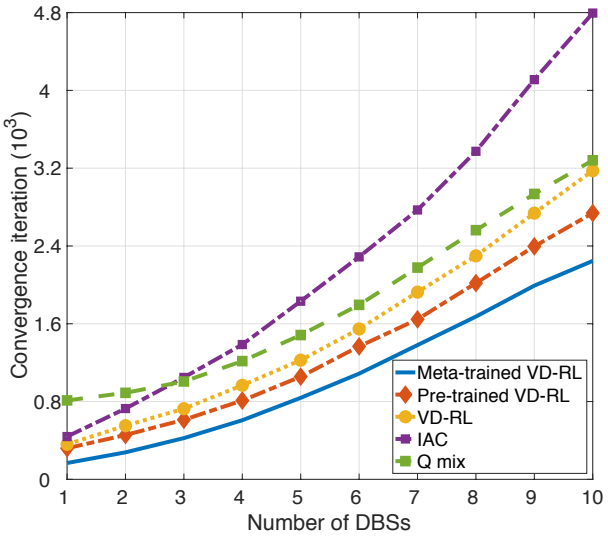
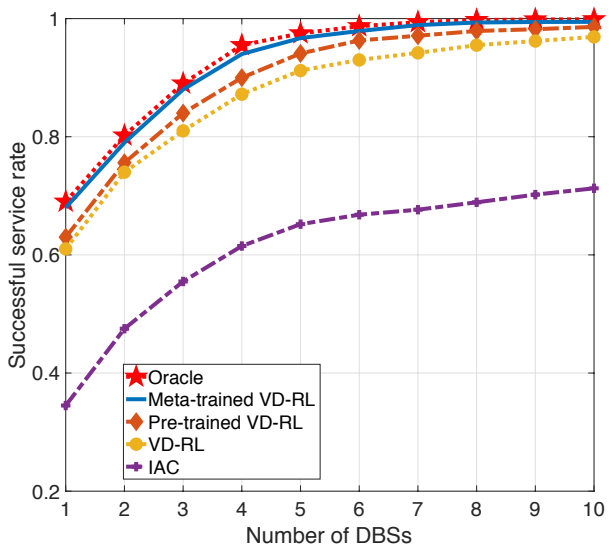


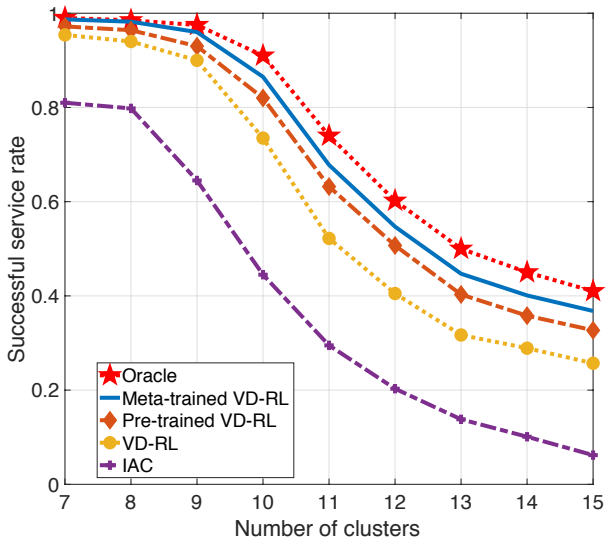
Fig. 11. Convergence of the meta-trained VD-RL algorithm as the number of DBSs varies. From this figure, we observe that the meta trained initialization has effectively reduce the complexity of the VD-RL algorithm.

initialization, the DBSs can find a team optimal strategy within a small number of policy update steps. The pre-trained VD-RL, however, converges to a much lower successful service rate, since it could make the DBSs to start their policy updating procedure from the initializations that are far from the optimal policy and value functions.

Fig. 11 shows the convergence of the proposed meta-training method as the number of DBSs varies. Fig. 11 shows that, as the number of DBSs increases, the number of iterations needed for convergence increases because of the associated growth in the size of the action and state spaces. From Fig. 11, we observe that, as the number of DBSs increases, the number of iterations needed for convergence will grow at a higher speed. This is because the action and state spaces within the problem increase exponentially with the number of DBSs. This is consistent with the complexity analysis in Section III. A. Moreover, the number of iterations that the Q mix and the proposed VD-RL algorithm need for convergence increases much slower compared to the traditional IAC algorithm. This is because the value decomposition scheme reduces the dimensionality of the action and state spaces in the considered problem. In particular, the value decomposition scheme enables each DBS to update its strategy using its own actions and states, thus simplifying the considered problem. However, the neural networks deployed for value decomposition within the Q mix algorithm introduces extra complexity at the DBSs, especially in simpler system as shown in Fig. 11. The convergence speed of the meta-trained VD-RL scheme decreases relatively slower compared to the original VD-RL algorithm, since the initializations of the policy and value function parameters in the meta-trained VD-RL algorithm have optimized performance across diverse user access request realizations. Moreover, the pre-trained VD-RL algorithm only slightly improves the VD-RL algorithm's convergence speed,



(a)



(b)

Fig. 12. Successful service rate as (a) the number of DBSs varies, (b) the number of clusters varies.

as the initialization of pre-trained VD-RL algorithm could be far from the optimal policy and value functions in unseen environments.

Fig. 12 shows how the successful service rate achieved by the DBSs varies with the number of DBSs and the number of clusters. From Fig. 12, we can see that the successful service rate provided by the DBSs increases with the number of deployed DBSs, but it decreases with the number of active clusters. This is because a larger number of DBSs can lead to a better coverage to the ground users. Fig. 12 also shows that the successful service rate increases more slowly when the number of deployed DBSs increases, or when the number of clusters is smaller. This is because the number of covered user requests increases when there are more DBSs available. Fig. 12 shows that the successful service rate also decreases more slowly

when the number of clusters in the simulated area is over 11. This is because the SLF time of the DBSs decreases when we have more clusters. From Fig. 12, we can also see that the meta-trained VD-RL algorithm yields a better coverage to the ground users, in particular, it yields a successful service rate that is 5.6% higher than the one resulting from the pre-trained VD-RL algorithm, 9.2% higher than the one resulting from the VD-RL algorithm, and 53.2% higher than the one resulting from the IAC algorithm. This is due to the fact that, by reducing the losses defined in (12) and (13) across various environments, the proposed meta-training method can find a set of policy and value functions that are close to the optimal policy and value functions at all possible user requests in $p(\mathcal{Z})$. This guarantees a faster convergence to a team optimal strategy in an unseen environment. Note that, although the meta-training method only yields small improvements to the successful service rate, it can converge much faster than the VD-RL algorithm.

V. Conclusion

In this paper, we have studied the problem of trajectory design for a group of DBSs in unpredictable, dynamic environments. In the considered system, the DBSs cooperatively fly around the considered environment to provide on demand uplink communication service to ground users. We have formulated the studied problem in an optimization setting and have proposed a VD-RL algorithm to solve this problem. The proposed VD-RL algorithm makes the DBSs independently update their individual strategies toward the maximal coverage in the system, by sharing only its utility and value to other DBSs. To improve the convergence speed of the VD-RL algorithm in unseen environments, we have also proposed a meta-training method to optimize the initializations in a VD-RL solution. Simulation results show that the proposed VD-RL algorithm with meta training mechanism outperforms the traditional MARL algorithms.

Appendix

A. Proof of Proposition 1

Proof. To prove the convergence of the proposed VD-RL algorithm, we only need to prove that the proposed VD-RL algorithm satisfies the following conditions in [26]:

1) Value function $V(\boldsymbol{S}_k)$ converges to a local minimum value of $A^2(\boldsymbol{a}_k, \boldsymbol{S}_k)$, that is
$$2\sum_{k=1}^{K} A(\boldsymbol{a}_k, \boldsymbol{S}_k)\nabla_{\boldsymbol{\theta}_c} A(\boldsymbol{a}_k, \boldsymbol{S}_k) = 0. \quad (15)$$
2) Policy function parameters are updated as in
$$\boldsymbol{\theta}_a^{(i+1)} = \boldsymbol{\theta}_a^{(i)} + \alpha_a^{(i)} \sum_{k=1}^{K} A(\boldsymbol{a}_k, \boldsymbol{S}_k)\nabla_{\boldsymbol{\theta}_a} \log \pi_{\boldsymbol{\theta}_a}(\boldsymbol{a}_k | \boldsymbol{S}_k). \quad (16)$$
3) $\max_{\boldsymbol{a}_{n,k}\in\mathcal{C}\cup\{O\}, \boldsymbol{s}_{n,k}\in\mathcal{S}} \left| \frac{\partial \pi_{\boldsymbol{\theta}_a}(\boldsymbol{a}_k|\boldsymbol{S}_k)}{\partial\theta_i \partial\theta_j} \right| < \infty$, for any elements θ_i and θ_j in the policy function parameter vector $\boldsymbol{\theta}_a$.
4) $\lim_{i\to\infty} \alpha_c^{(i)} = 0$, $\sum_{i=1}^{\infty} \alpha_c^{(i)} = \infty$, $\lim_{i\to\infty} \alpha_a^{(i)} = 0$, and $\sum_{i=1}^{\infty} \alpha_a^{(i)} = \infty$.

Next, we prove that the proposed VD-RL algorithm satisfies condition 1). When value function $\tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k})$ converges to a local minimum, such that
$$\begin{aligned} &2A(\boldsymbol{a}_k, \boldsymbol{S}_k)\nabla_{\boldsymbol{\theta}_{c,n}} \tilde{V}_{\boldsymbol{\theta}_{c,n}}(\boldsymbol{s}_{n,k}) \\ &= 2A(\boldsymbol{a}_k, \boldsymbol{S}_k)\nabla_{\boldsymbol{\theta}_{c,n}} \tilde{A}_n(a_{n,k}, \boldsymbol{s}_{n,k}) = 0. \end{aligned} \quad (17)$$
with the assumption $A(\boldsymbol{a}_k, \boldsymbol{S}_k) \neq 0$, we have $\nabla_{\boldsymbol{\theta}_{c,n}} \tilde{A}_n(a_{n,k}, \boldsymbol{s}_{n,k}) = 0$. That is, we have $2\sum_{k=1}^{K} \tilde{A}_n(a_{n,k}, \boldsymbol{s}_{n,k})\nabla_{\boldsymbol{\theta}_{c,n}} \tilde{A}_n(a_{n,k}, \boldsymbol{s}_{n,k}) = 0$. Thus, VD-RL algorithm satisfies condition 1).

Next, we can see that the VD-RL's distributed update on policy function parameters satisfies condition 2), as each DBS n updates its policy function parameters in the form of $\boldsymbol{\theta}_{a,n}^{(i+1)} = \boldsymbol{\theta}_{a,n}^{(i)} + \alpha_a^{(i)} \sum_{k=1}^{K} \tilde{A}_n(a_{n,k}, \boldsymbol{s}_{n,k})\nabla_{\boldsymbol{\theta}_{a,n}} \log \pi_{\boldsymbol{\theta}_{a,n}}(a_{n,k} | \boldsymbol{s}_{n,k})$. Thus, the distributed update on policy parameters $\boldsymbol{\theta}_{a,n}$ satisfies condition 2). Condition 3) can be satisfied by properly setting neural network of the policy functions in the proposed VD-RL algorithm (e.g. setting activation function). Meanwhile, condition 4) can be satisfied by adjusting the step sizes of the policy and value functions. In consequence, the proposed algorithm implemented at each DBS n satisfies all conditions from 1) to 4). In other words, each DBS n is guaranteed to converge to an optimal strategy $\boldsymbol{\pi}_n^*$ that yields a local maximal team utility $\overline{G}(\boldsymbol{\pi}_n^*, \boldsymbol{\pi}_{-n})$, with DBSs $n' \in \mathcal{N} \setminus n$ following strategies in $\boldsymbol{\pi}_{-n} = [\boldsymbol{\pi}_n]_{n'\in\mathcal{N}\setminus n}$. In summery, by updating policies at each DBS in the system, the proposed VD-RL algorithm solves problem (7) step by step in the form of
$$\begin{aligned} \max_{\boldsymbol{\pi}_1} \ldots \max_{\boldsymbol{\pi}_N} &\sum_{\boldsymbol{\xi}\in\mathcal{E}} G(\boldsymbol{\xi}) \prod_{k=1}^{K} \pi_1(\xi | \xi_{1,k}, \tau_{1,k}) \prod_{k=1}^{K} \pi_2(\xi | \xi_{2,k}, \tau_{2,k}) \\ &\ldots \prod_{k=1}^{K} \pi_N(\xi | \xi_{N,k}, \tau_{N,k}) \end{aligned} \quad (18)$$
$$\text{s. t.} \quad \sum\nolimits_{\boldsymbol{\xi}\in\mathcal{E}} \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_n(\xi | \xi_{n,k}, \tau_{n,k}) = 1, \quad (18a)$$
$$\sum\nolimits_{a_{n,k}\in\mathcal{C}\cup\{O\}} \pi_n(\xi | \xi_{n,k}, \tau_{n,k}) = 1, \forall n \in \mathcal{N}, \boldsymbol{\xi} \in \mathcal{E}, k \in \mathcal{K}, \quad (18b)$$
$$0 \leq \pi_n(\xi | \xi_{n,k}, \tau_{n,k}) \leq 1, \forall n \in \mathcal{N}, \boldsymbol{\xi} \in \mathcal{E}, k \in \mathcal{K}, \quad (18c)$$
As the DBSs' strategies are independent, the local optimal strategy at each DBS n constructs a local optimal strategy of the non-convex problem (7). This completes the proof. □

References

[1] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Meta-reinforcement learning for trajectory design in wireless UAV networks," in *in Proc Global Communications Conference (GLOBECOM)*, Taipei, Taiwan, Dec. 2020.

[2] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3039–3071, Fourthquarter 2019.

[3] F. Jiang and A. L. Swindlehurst, "Optimization of UAV heading for the ground-to-air uplink," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 993–1005, 2012.

[4] M. Mozaffari, A. Taleb Zadeh Kasgari, W. Saad, M. Bennis, and M. Debbah, "Beyond 5G with UAVs: Foundations of a 3D wireless cellular network," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 357–372, 2019.

[5] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, June 2017.

[6] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, "Optimal 3D-trajectory design and resource allocation for solar-powered UAV communication systems," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4281–4298, Feb. 2019.

[7] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, Jan. 2018.

[8] Z. M. Fadlullah, D. Takaishi, H. Nishiyama, N. Kato, and R. Miura, "A dynamic trajectory control algorithm for improving the communication throughput and delay in UAV-aided networks," *IEEE Network*, vol. 30, no. 1, pp. 100–105, Jan. 2016.

[9] M. Hua, A. L. Swindlehurst, C. Li, and L. Yang, "UAV-aided backscatter networks: Joint UAV trajectory and protocol design," in *in Proc Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019.

[10] Y. Huang, X. Mo, J. Xu, and L. Qiu, "Reinforcement learning for maneuver design in UAV-enabled NOMA system with segmented channel," *arXiv preprint arXiv:1908.03984*, Aug. 2019.

[11] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, Aug. 2018.

[12] A. Ferdowsi, M. A. Abd-Elmagid, W. Saad, and H. S. Dhillon, "Neural combinatorial deep reinforcement learning for age-optimal joint trajectory and scheduling design in UAV-assisted networks," *IEEE Journal on Selected Areas on Communications (JSAC), Special Issue on Age of Information in Real-time Systems and Networks, to appear*, 2021.

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

[14] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, to appear, 2020.

[15] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 729–743, Aug. 2020.

[16] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7957–7969, May 2019.

[17] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *arXiv preprint arXiv:1705.08926*, 2017.

[18] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, 2018.

[19] R. Houthooft, Y. Chen, P. Isola, B. Stadie, F. Wolski, O. J. Ho, and P. Abbeel, "Evolved policy gradients," in *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, Dec 2018, pp. 5400–5409.

[20] S. Ritter, J. X Wang, Z. Kurth-Nelson, S. M Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick, "Been there, done that: Meta-learning with episodic recall," *arXiv preprint arXiv:1805.09692*, 2018.

[21] Z. Xu, H. P van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, Dec. 2018.

[22] D. W. K. Ng, E. S. Lo, and R. Schober, "Energy-efficient resource allocation in OFDMA systems with large numbers of base station antennas," *IEEE Transactions on Wireless Communications*, vol. 11, no. 9, pp. 3292–3304, Jul 2012.

[23] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *IEEE Global Communications Conference*, Austin, USA, Dec 2014, pp. 2898–2904.

[24] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[25] 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 Transactions on Wireless Communications*, to appear, 2020.

[26] R. S Sutton, D. A McAllester, S. P Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, Denver, USA, Dec. 2000.

[27] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z Leibo, K. Tuyls, et al., "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. of International Conference on Machine Learning (ICML)*, Sydney, Australia, Aug. 2017.

[29] L. C. Baird III, "Advantage updating," Tech. Rep., WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.

[30] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[31] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, Apr 2012.