Swarm UAV Networking With Collaborative Beamforming and Automated ESN Learning in the Presence of Unknown Blockages

Sabarish Krishna Moorthy[a*], Nicholas Mastronarde[a], Scott Pudlewski[b],
Elizabeth Serena Bentley[c] and Zhangyu Guan[a]
[a]Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260, USA
[b]Georgia Tech Research Institute (GTRI), Atlanta, GA 30332, USA
[c]Air Force Research Laboratory (AFRL), Rome, NY 13440, USA
Email: {sk382, nmastron, guan}@buffalo.edu, scott.pudlewski@gtri.gatech.edu, elizabeth.bentley.3@us.af.mil

Abstract

This paper aims at designing high-data-rate swarm UAV networks with distributed beamforming capabilities. The primary challenge is that the beamforming gain in swarm UAV networks is highly affected by the UAVs' flight altitude, their movements and the resulting intermittent link blockages, as well as the availability of channel state information (CSI) at individual UAVs. To address this challenge, we propose *FlyBeam*, a learning-based framework for joint flight and beamforming control in swarm UAV networks. We first present a mathematical formulation of the control problem with the objective of maximizing the throughput of swarm UAV networks by jointly controlling the flight and distributed beamforming of UAVs. Then, a distributed solution algorithm is designed based on a combination of Echo State Network (ESN) learning and online reinforcement learning. The former is adopted to approximate the utility function for individual UAVs based on online measurements, by jointly considering the unknown blockage dynamics and other factors that affect the beamforming gain. The latter is used to guide the exploitation and exploration in *FlyBeam*. We further design a scheme referred to as *AutoESN* to automate the training of the ESN model. *AutoESN* can update the configurable ESN parameters automatically using a combination of loss function and step size. The effectiveness of *FlyBeam* is evaluated through an extensive simulation campaign on UBSim, a Python-based Universal Broadband Simulator for integrated aerial and ground wireless networking. The performance of *FlyBeam* is compared with two benchmark schemes, one designed based on traditional optimization techniques and the other based on learning with utility function approximation through Long Short-Term Memory (LSTM). In the performance evaluation, we consider both Zero-Forcing (ZF) and Maximum Ratio Transfer (MRT) for beamforming with sequential and simultaneous channel state estimation. It is shown that significant (up to 450%) beamforming gain can be achieved by *FlyBeam*. We also investigate the effects of blockages and UAV flight altitude on the beamforming gain. It is found that, somewhat surprisingly, higher *(rather than lower)* beamforming gain can be achieved by *FlyBeam* with denser blockages in swarm UAV networks.

Keywords: Swarm UAV Networks, Distributed Beamforming, Echo State Network, Reinforcement Learning.

*Corresponding author

[1]A preliminary shorter version of this paper appeared in the Proceedings of IEEE International Conference on Communications (ICC), Virtual/Montreal, Canada, 2022 [1].

[2]ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER: (a) Contractor acknowledges Government's support in the publication of this paper. This material is based upon work funded by AFRL, under AFRL Contract No. FA8750-20-1-0501 and FA8750-20-C-1021, and in part by the NSF under Grant SWIFT-2229563. (b) Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

[3]Distribution A. Approved for public release: Distribution unlimited AFRL-2023-0492 on 30 Jan 2023.

1. Introduction

Unmanned aerial vehicles (UAVs) have been envisioned as an enabling technology for a wide set of new applications, because of their features of fast deployment, high mobility and small size [2–5]. Examples of these applications include small cells with flying base stations, UAV-aided guidance, swarm networking for field sensing and data collection, emergency wireless networking in the aftermath of disasters, among others. While UAVs can certainly enable a wide set of new applications, their wide deployment will impose a significant burden on the capacity of the underlying wireless networks. In this work, we focus on designing high-data-rate wireless UAV networks by exploring spatial diversity through collaborative beamforming among the UAVs.

Since it is not easy to mount many antennas on individual UAVs because of their small size, in this work we consider a swarm of UAVs collaborating with each other to perform distributed beamforming. One of the primary challenges is in the formation of UAV clusters for collaborative beamforming. In swarm UAV networks, beamforming can be typically accomplished in two phases. In the first phase, the UAVs estimate the channel state information (CSI) of the wireless links from them to the users they serve and then share the obtained CSI among the UAVs; in the second phase, the UAVs collaborate with each other to perform distributed beamforming for data transmission. While higher beamforming gain can be achieved by forming a larger UAV cluster with more antennas, it takes longer for the CSI sharing in the first phase and hence reduces the time available for data transmission in the second phase. Therefore, a tradeoff needs to be achieved between beamforming gain and channel utilization. Moreover, the beamforming gain is closely coupled with the statistical behaviors of the wireless channels, which are affected by the flight altitude of the UAVs, the dynamic movements of UAVs and the resulting intermittent existence of blockages. Roughly speaking, non-line-of-sight (NLOS) transmissions decrease and line-of-sight (LOS) transmission increase as UAVs fly higher. Additionally, the interference level in the network can be effectively lowered with more and larger blockages, and this also affects the beamforming gain.

To account for these coupled factors that jointly affect the beamforming gain in swarm UAV networks, in this paper we propose *FlyBeam*, a learning-based framework for joint flight and beamforming control in swarm UAV networks with unknown blockage dynamics. The main contributions of the paper are as follows.

- ***FlyBeam Control Problem Formulation.*** We first present a mathematical formulation of the *FlyBeam* control problem, where the objective is to maximize the aggregate capacity of swarm UAV networks with a set of single-antenna UAVs collaborating with each other to perform distributed beamforming. Two channel estimation schemes are considered in the formulation, namely sequential and simultaneous channel estimations.

- ***Distributed Solution Algorithm Design.*** We then design a distributed solution algorithm to solve the *FlyBeam* control problem based on a combination of Echo State Network (ESN) learning and reinforcement learning (RL), where the former provides an approximation of the utility functions of the UAVs with unknown blockage dynamics based on online measured data, and the latter is adopted to achieve a good tradeoff between exploitation and exploration in *FlyBeam*. Furthermore, we propose an iterative ESN training approach, based on which data samples are collected at network run time and are used to re-train the initially trained ESN model to improve the overall prediction accuracy.

- ***Automated Tuning of ESN Parameters.*** In order to automate the training the ESN model we design a scheme referred to as *AutoESN*. *AutoESN* can update the configurable ESN parameters automatically using a combination of loss function and step size. This scheme can enable faster and more accurate tuning of the meta parameters of the ESN model with reduced human intervention.

- ***Experimental Evaluation.*** We evaluate the effectiveness of *FlyBeam* by conducting an extensive simulation campaign over a Python-based simulator called UBSim, a Universal Broadband Simulator for integrated aerial and ground wireless networking. First, we analyse the performance of FlyBeam and compare it with a benchmark with utility function approximation through Long Short-Term Memory (LSTM) learning. Then we show the effectiveness of *AutoESN* for automated tuning of ESN meta parameters. We further analyse the performance of *FlyBeam* by comparing it with a benchmark scheme based on traditional optimization techniques. We also investigate the effects of blockages and UAV flight altitude on the beamforming gain considering both ZF and MRT beamforming with sequential and simultaneous channel estimation.

To the best of our knowledge, *FlyBeam* is the first RL framework for swarm UAV networks with iterative ESN-based utility function approximation jointly considering UAV movement, blockage- and altitude-dependent wireless channels, as well as CSI-sharing in distributed beamforming with network run-time feedback to improve the prediction accuracy of the distributed solution algorithm.

The remainder of this paper is organized as follows. We discuss the related work in Section 2. In Section 3, we describe the system model and problem formulation. In Section 4, we describe the design of *FlyBeam*. The performance evaluation results are discussed in Section 5, and finally we draw the main conclusions in Section 6.

2. Related Work

Wireless UAV networking has drawn significant research attention over the past years [6–11]. For example, in [6] Azari et al. study power control for wireless communications among single-antenna UAVs in cellular networks. In [7], the authors maximize the throughput in UAV-enabled orthogonal frequency-division multiple access (OFDMA) systems with delay-constrained data traffic. The authors of [8] jointly optimize the trajectory and communication in multi-UAV wireless networks to achieve better fairness among users. In [9], we propose a new framework for automated control of swarm UAV networks based on recent results on principled software-defined wireless networking. Readers are referred to [10, 11] and references therein for a good survey of the main results in this area. *Different from these works, which focus on non-collaborative single-antenna UAV communications, in this paper we explore spatial diversity in swarm UAV networks by allowing the UAVs to perform distributed beamforming.*

Machine learning techniques such as Echo State Learning [12–15], Long Short-Term Memory [16–20] and RL [21–24] have been used as a possible solution to different network problems. For example, in [12], the authors use ESN to predict the future trajectories of user equipment to enable dynamic repositioning of UAV base stations. In [13], Liu et al. use ESN-based prediction algorithm to predict the future positions of users based on the real dataset. In [14], the authors use ESN to allocate resources for virtual reality (VR) users communicating using an UAV-enabled LTE over unlicensed (LTE-U) network. In [15], Liu et al. propose an ESN based prediction algorithm to predict the future positions of users based on anonymous user-trajectories in the physical world. In [16], the authors propose a deep RL (DRL) based flight resource allocation framework leveraging LSTM to predict network dynamics resulting from time-varying airborne channels and energy arrivals at the ground devices. In [17], Lin et al. adopt a combination of DRL and LSTM to accelerate the convergence speed of the dynamic spectrum interaction algorithm for UAV communications. In [18], the authors propose an LSTM-based deep learning location-aware predictive beamforming scheme to track the beam for UAV communications in dynamic scenarios. In [19], Yao et al. design a deep LSTM model for real-time path planning in unknown environments. The authors of [21] merge Deep Q Learning based DRL with massive MIMO to optimize the UAV navigation. In [22], Cui et al. develop a multi-agent RL algorithm for resource allocation in UAV networks. In [23], the authors develop an interference-aware path planning scheme using DRL algorithm based on ESN for cellular-connected UAVs. In [24], Wang et al. propose an RL-based user association and resource allocation algorithm for multi-UAV enabled mobile edge computing (MEC) applications. *Unlike these efforts, in this work we propose an online learning solution called FlyBeam based on the combination of iterative ESN-based RL control for the UAV swarm control problem considering the factors that affect the beamforming gain in swarm UAV networks with unknown blockage dynamics.*

Distributed beamforming has also been extensively studied in wireless networks [25–34]. For example, in [25], Mohanti et al. propose an SDR-based experimental framework for UAV networks to assign beamforming weights to ensure high level of directivity. In [26], the authors study beamforming vector generation and updating based on recursive channel estimation. The beamforming algorithms for UAV swarm are studied in [27] where the swarm is modeled as a morphing volumetric random array. However, unlike our work, [27] did not consider the dynamic movements of swarm UAVs and the effects of blockages on the beamforming gain. The authors of [28] present a cooperative communication scheme for cache-enabled UAVs to jointly decide the UAV placement and transmit beamforming based on outdated CSI information. Similar to [27], [28] did not consider the effect of blockages either. In [29], Yuan et al. consider a single UAV-user pair and develop a deep learning-based predictive beamforming scheme that can recover from beam misalignment caused by UAV jittering. Position-based beamforming is studied in [30] to enhance the capacity of UAV communications in LTE networks in the presence of direction-of-arrival estimation errors. In [31], the authors discuss the feasibility and enabling techniques for distributed beamforming in swarm UAV networks. In our previous work [32], we design distributed algorithms for joint power, association and flight

Notation	Physical Meaning
$\mathcal{M}$	Set of single-antenna UAVs
$\mathcal{U}$	Set of ground users
$\mathcal{S}$	Set of blockages
L_s, W_s, H_s	Length, width, height of a blockage $s \in \mathcal{S}$
C_s, θ_s	Center and orientation of a blockage $s \in \mathcal{S}$
$\mathbf{P}_s^{\text{blk}}(C_s, L_s, W_s, H_s, \theta_s)$	Set of points contained in blockage $s \in \mathcal{S}$
$\mathbf{cod}_u$	Location vector of user $u \in \mathcal{U}$
$\mathbf{cod}_m$	Location vector of UAV $m \in \mathcal{M}$
$\mathbf{P}_{mu}^{\text{link}}$	Set of points on line connecting UAV m and user u
$\mathbf{P}_{mus}^{\text{blk}}$	Set of intersection points
S_{mu}	Total number of blockages on the link $[u, m]$
$\mathrm{I}(\cdot)$	Indicator function
f	Operating frequency band
$H_{mu}(f)$	Path loss for link $[u, m]$
C	Speed of light
η_0	Per-blockage absorption coefficient
d_{mu}	Distance between UAV m and user u
$\beta_{mu}(f)$	Path-loss exponent
K	Rician Factor
h_{mu}	Channel fading coefficient
$\mathbf{y}_m$	Received pilot signal by UAV $m \in \mathcal{M}$
$\boldsymbol{p}_u$	Pilot sequence of user $u \in \mathcal{U}$
n_m^0	Vector of Additive White Gaussian Noise (AWGN)
$\widetilde{h}_{mu}$	Estimated channel gain of link [u,m]
t_{est}	Time overhead for the channel estimation for user
N_{plt}	Number of bits in each pilot sequence
r	Data rate for pilot transmission
$\boldsymbol{w}$	Beamforming weight vector
t_{csi}	Time overhead for CSI sharing
t_{beam}	Time overhead for beamforming weight feedback
t_{ovhd}	Overall time overhead
γ	Channel utilization coefficient
t_{slt}	Duration of each time slot
N_u^0	Noise power at ground user u
$\mathcal{T}$	Total network running time
ρ_m^t	Input to ESN module
$\Gamma_m'^t$	Index of UAV m''s rectangle in time t
ξ_m^t	Set of actions for UAV m
$\mathcal{Q}$	Set of sub-channels

Table 1: Summary of Key Notations.

control in swarm UAV networks with each UAV endowed with a large number of antennas. Please refer to [33, 34] and references therein for an extensive survey of the latest results in this area. *None of these works have studied distributed beamforming in swarm UAV networks by explicitly considering all the factors that affect the beamforming gain discussed in Section 1, including the flight of UAVs, the resulting dynamic blockages, and CSI-sharing among UAVs.*

3. System Model and Problem Formulation

We consider swarm UAV networks with a set $\mathcal{M}$ of single-antenna UAVs collaborating with each other to serve a set $\mathcal{U}$ of ground users. Here, we consider pre-clustered swarm UAVs. The clustering of UAVs can be achieved by using swarm-intelligence-based localization and clustering schemes [35], mobility and location-aware stable clustering [36], among others. The UAVs are allowed to form a virtual MIMO UAV cluster to enhance the quality of the aerial-ground wireless links through distributed beamforming. Focusing on the downlink communications in this setting, our objective is to investigate the effects of those factors that affect the beamforming gain, including blockages in the network, flight altitude of the UAVs as well as the altitude-dependent fading channels, among others. The

results obtained in this paper can also be extended to uplink scenarios. Next we describe the blockage, channel and beamforming models sequentially.

3.1. Blockage Model

Denote $\mathcal{S}$ and $|\mathcal{S}|$ as the set and the number of blockages in the network, respectively. For each blockage $s \in \mathcal{S}$, let C_s, L_s, W_s, H_s and θ_s represent the center, length, width, height and orientation of the blockage, respectively. The orientation θ_s is considered to be uniformly distributed in $[0,\ 2\pi]$, and the other blockage parameters are randomly generated with the average corresponding to the typical size of blockages in real networks, e.g., the buildings. Let $\mathbf{P}_s^{\text{blk}}(C_s, L_s, W_s, H_s, \theta_s)$ represent the set of points contained in blockage $s \in \mathcal{S}$.

Denote the location vector of user $u \in \mathcal{U}$ as $\mathbf{cod}_u = (x_u,\ y_u,\ z_u)$, with $x_u,\ y_u,\ z_u$ representing the x-, y- and z-axis coordinates, respectively. Similarly, denote $\mathbf{cod}_m = (x_m,\ y_m,\ z_m)$ as the location vector of UAV $m \in \mathcal{M}$. Let $\mathbf{P}_{mu}^{\text{link}}$ denote the set of points on the line connecting UAV $m \in \mathcal{M}$ and user $u \in \mathcal{U}$. Further denote $\mathbf{P}_{mus}^{\text{xt}} = \mathbf{P}_{mu}^{\text{link}} \cap \mathbf{P}_s^{\text{blk}}(C_s, L_s, W_s, H_s, \theta_s)$ as the resulting intersection set of points. Then, given the set $\mathcal{S}$ of blockages, the total number of blockages on the link between user u and UAV m, denoted as S_{mu}, can be expressed as

$$S_{mu} = \sum_{s \in \mathcal{S}} \mathrm{I}(\mathbf{cod}_m, \mathbf{cod}_u, s),\ \forall m \in \mathcal{M},\ \forall u \in \mathcal{U}, \tag{1}$$

where

$$\mathrm{I}(\mathbf{cod}_m, \mathbf{cod}_u, s) = \begin{cases} 1, & \text{if } \mathbf{P}_{mus}^{\text{xt}} \neq \phi \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

is the indicator function taking the value of 1 if blockage s is blocking the link and 0 otherwise.

3.2. Channel Model

Denote $H_{mu}(f)$ as the path loss for the link between UAV $m \in \mathcal{M}$ and ground user $u \in \mathcal{U}$ operating in frequency band $f \in \mathcal{F}$. Then we model $H_{mu}(f)$ as in [37] as follows:

$$H_{mu}(f) = \eta_0^{S_{mu}} \left(\frac{4\pi f}{C}\right)^2 (d_{mu})^{\beta_{mu}(f)}, \tag{3}$$

where C is the speed of light, $\beta_{mu}(f)$ is the path-loss exponent for the link between UAV $\in \mathcal{M}$ and ground user $u \in \mathcal{U}$ in frequency band f, S_{mu} defined in (1) represents the number of blockages in the link, $\eta_0 \in [0,\ 1]$ is the per-blockage absorption coefficient [38] [4], and finally $d_{mu} = d_{mu}(\mathbf{cod}_m, \mathbf{cod}_u)$ denotes the distance between UAV $m \in \mathcal{M}$ and user $u \in \mathcal{U}$.

The transmission time is divided into a set of consecutive time slots. We consider block fading channels in each time slot, i.e., the channel coefficient is considered to be fixed in each time slot and change to another random value following certain distribution in the next. The Rician fading model is adopted to characterize the fading behavior of the wireless channels, with the Rician factor K depending on whether the link is blocked or not. For NLOS links, i.e. $S_{mu} \neq 0$ in (3), factor K is set to 0; for LOS transmissions, i.e., $S_{m,u} = 0$, factor K is given as $K = 13 - 0.03 \times d_{mu}$ [39]. Denote the resulting channel fading coefficient as h_{mu} for the wireless link between UAV $m \in \mathcal{M}$ and ground user $u \in \mathcal{U}$.

3.3. Channel Estimation Model

We consider pilot-based channel estimation, which can be accomplished in either a sequential or simultaneous manner. In the former case, the ground users broadcast their pilot signals in each time slot sequentially, and hence the

[4] The per-blockage absorption coefficient is used to take into account the attenuation of signal after passing though a blockage. For example, if signal x passes through a blockage that has an absorption coefficient of η_0, then the $\eta_0 \times x$ of signal will be absorbed allowing $1 - \eta_0 \times x$ of signal pass through the blockage. Again, if this signal again passes through another blockage of same absorption coefficient η_0, then the resulting signal will be $\eta_0 \times \eta_0 \times x$ which is $\eta_0^2 \times x$, and so on.

problem of pilot contamination can be avoided but at the cost of more estimation time. In contrast, with simultaneous channel estimation all the users broadcast their pilot signals at the same time, and hence it takes less time at the cost of lower estimation accuracy. Next, we describe the two estimation techniques briefly.

Sequential Channel Estimation. Denote $\mathbf{y}_m^{\text{seq}} = [y_{m1}^{\text{seq}}, \cdots, y_{mN_{\text{plt}}}^{\text{seq}}]$ as the pilot signal received by UAV $m \in \mathcal{M}$, with $y_{mv},\ v = 1, \cdots, N_{\text{plt}}$ being the vth symbol of the received pilot signal. Then $\boldsymbol{y}_m^{\text{seq}}$ can be given as

$$\boldsymbol{y}_m^{\text{seq}} = \sqrt{H_{mu}} h_{mu} \boldsymbol{p}_u + n_m^0, \tag{4}$$

where $\boldsymbol{p}_u = [p_{u1}, \cdots, p_{uN_{\text{plt}}}]$ is the pilot sequence of user $u \in \mathcal{U}$, and $n_m^0 = (n_{mv}^0)_{v=1}^{N_{\text{plt}}}$ is the vector of Additive White Gaussian Noise (AWGN) at UAV $m \in \mathcal{M}$. Let $\widetilde{h}_{mu}$ represent the estimated channel gain for the link between UAV $m \in \mathcal{M}$ and user $u \in \mathcal{U}$. Then, if a least-square estimator [40] is considered, we have

$$\widetilde{h}_{mu}^{\text{seq}} = \boldsymbol{y}_m^{\text{seq}} \times \boldsymbol{p}_u^{\dagger} (\boldsymbol{p}_u \times \boldsymbol{p}_u^{\dagger})^{-1}, \tag{5}$$

where $(\cdot)^{\dagger}$ denotes the conjugate transpose and $(\cdot)^{-1}$ represents the matrix inverse operation.

Let $t_{\text{est},u}$ denote the time overhead for the channel estimation for user $u \in \mathcal{U}$ and t_{est} as the total time overhead for all the users in $\mathcal{U}$. Then we have

$$t_{\text{est}}^{\text{seq}} = \sum_{u \in \mathcal{U}} t_{\text{est},u}. \tag{6}$$

where $t_{\text{est},u} = N_{\text{plt}}/r$ with N_{plt} being the number of bits in each pilot sequence and r the data rate for pilot transmission.

Simultaneous Channel Estimation. In Simultaneous channel estimation, all the users broadcast their pilot signals at the same time which takes less estimation time however will degrade the estimation accuracy because of pilot contamination. For a large-scale network with pilot reuse, pilot decontamination can be achieved by using orthogonal pilot sequences with time-shifted protocol (TSP) for pilot transmission [41], combination of TSP with power allocation algorithms [42], covariance based channel estimation[43], among others. In this work, we consider the simple yet effective orthogonal pilot sequence based channel estimation, while this work can also be extended to other pilot decontamination techniques [44, 45]. Denote $\mathbf{y}_m^{\text{sim}} = [y_{m1}^{\text{sim}}, \cdots, y_{mN_{\text{plt}}}^{\text{sim}}]$ as the pilot signal received by UAV $m \in \mathcal{M}$, with $y_{mv},\ v = 1, \cdots, N_{\text{plt}}$ being the vth symbol of the received pilot signal. Then the received pilot signal $\boldsymbol{y}_m^{\text{sim}}$ can be expressed as

$$\boldsymbol{y}_m^{\text{sim}} = \sum_{u \in \mathcal{U}} \sqrt{H_{mu}} h_{mu} \boldsymbol{p}_u + n_m^0, \tag{7}$$

and the estimated channel gain can be expressed similar to (5). Denote the resulting time overhead for simultaneous channel estimation as $t_{\text{est}}^{\text{sim}}$.

3.4. Beamforming Model

Each UAV $m \in \mathcal{M}$ sends its CSI obtained above to a pre-selected leading UAV of the swarm, denoted as m' with $m' \in \mathcal{M}/m$, which will then calculate the beamforming weights for the whole swarm. Denote t_{csi} as the resulting time overhead, then we have

$$t_{\text{csi}} = \sum_{m \in \mathcal{M}/m'} N_{\text{csi}}/C_{mm'} \tag{8}$$

where $C_{mm'}$ represents the capacity of the link between UAVs m and m', and N_{csi} is the amount of CSI data in bits to be shared by UAV $m \in \mathcal{M}$, i.e., $\widetilde{\boldsymbol{h}}_m = (\widetilde{h}_{mu})_{u \in \mathcal{U}}$ with $\widetilde{h}_{mu}$ obtained in (5). Here, we consider sequential transmission for CSI sharing. Based on the collected CSI, the beamforming weights can be calculated at the leading UAV. Denote the resulting beamforming weight vector as $\boldsymbol{w} = (\boldsymbol{w}_m)_{m \in \mathcal{M}}$, where $\boldsymbol{w}_m = (w_{mu})_{u \in \mathcal{U}}$ with w_{mu} being the beamforming weight of UAV m for ground user u. For example, based on *Zero Forcing (ZF)* beamforming [46], $\boldsymbol{w}_m$ can be given as

$$\boldsymbol{w}_m = \boldsymbol{h}_m^{\dagger} (\boldsymbol{h}_m \boldsymbol{h}_m^{\dagger})^{-1}, \tag{9}$$

where $(\cdot)^{\dagger}$ denotes the conjugate transpose and $(\cdot)^{-1}$ is the matrix inverse operation. The obtained beamforming weight is then sent back to the corresponding UAVs for the actual use in the following data transmission. Denote the resulting time overhead as t_{beam} and can be given as

$$t_{\text{beam}} = \sum_{m \in \mathcal{M}/m'} N_{\text{beam}}/C_{mm'} \tag{10}$$

where N_{beam} is the amount of beamforming data in bits to be shared by the leading UAV m' to other UAVs. Then, the overall time overhead denoted as t_{ovhd} can be written as

$$t_{\text{ovhd}} = t_{\text{est}} + t_{\text{csi}} + t_{\text{beam}}, \tag{11}$$

where t_{est}, t_{csi} and t_{beam} are the above defined time overhead for channel estimation, CSI sharing and beamforming weight feedback, respectively.

3.5. FlyBeam Control Problem

For a given frequency f, bandwidth B and coordinates $\mathbf{cod}_u, u \in \mathcal{U}$, as well as the set of blockages $\mathcal{S}$, the control objective of *FlyBeam* is to maximize the aggregate network capacity by jointly controlling the flight and beamforming of the UAVs, as formulated as follows.

$$\begin{aligned} &max\ \gamma B \sum_{u \in \mathcal{U}} \log_2\Big(1 + \frac{\sum\limits_{m \in \mathcal{M}} P_{mu}\psi_{mu}}{\sum\limits_{u' \in \mathcal{U}/u} \sum\limits_{m \in \mathcal{M}} P_{mu'}\psi_{mu'} + N_u^0}\Big), \\ &s.t.: \quad \sum_{u \in \mathcal{U}} P_{mu} \le P_{\max},\ \forall m \in \mathcal{M}, \end{aligned} \tag{12}$$

where $\psi_{mu} = \widehat{H}_{mu}|\widehat{h}_{mu}\widehat{w}_{mu}|^2$, $\widehat{H}_{mu} = H_{mu}(\mathbf{cod}_m)$, $\widehat{h}_{mu} = h_{mu}(\mathbf{cod}_m)$ and $\widehat{w}_{mu} = w_{mu}(\mathbf{cod}_m, \pi)$ denote the path loss, channel fading and beamforming weights for user u and UAV m, respectively, with π denoting the beamforming strategy; similarly $\psi_{mu'} = \widehat{H}_{mu'}|\widehat{h}_{mu'}\widehat{w}_{mu'}|^2$, $\widehat{H}_{mu'} = H_{mu'}(\mathbf{cod}_m)$, $\widehat{h}_{mu'} = h_{mu'}(\mathbf{cod}_m)$ and $\widehat{w}_{mu'} = w_{mu'}(\mathbf{cod}_m, \pi)$ denote the path loss, channel fading and beamforming weights for user u' and UAV m, respectively, with π denoting the beamforming strategy; P_{mu} and $P_{mu'}$ are the transmission power of UAV $m \in \mathcal{M}$ allocated to users u and u', respectively; $P_{\max}$ is the maximum transmission power of each UAV; N_u^0 is the power of noise at ground user $u \in \mathcal{U}$; and finally $\gamma = \frac{t_{\text{slt}} - t_{\text{ovhd}}}{t_{\text{slt}}}$ represents the channel utilization coefficient with t_{slt} being the duration of each time slot and t_{ovhd} defined in (11).

4. FlyBeam Algorithm Design

In (12), the optimization variable is $w_{mu} = w_{mu}(\mathbf{cod}_m, \pi)$ which is the beamforming weights for user u and UAV m with π denoting the beamforming strategy. Here, the UAV coordinate is controlled using FlyBeam which in turn affects the beamforming strategy. The primary challenge in solving problem (12) is that the wireless channel hence the beamforming weight w_{mu} and the channel utilization coefficient γ are closely coupled with the UAV location variables $\mathbf{cod}_m$ and hence the resulting dynamic blockages, for which the complete information is unknown to the UAVs. To address this challenge, in this work we solve the problem by designing distributed control algorithms based on a combination of ESN learning [47] and RL techniques. In this work, we optimize the flight control of the swarm UAVs while the UAVs are collaboratively serving the users following given beamforming strategy. The overall architecture of FlyBeam framework is shown in Fig. 1. It consists of two main modules, namely the *ESN Module* and the *RL module*. Particularly, the ESN is used to approximately model the mapping from the input signals to the output signals of a system, by training its input weights $\mathbf{W_{in}}$, the reservoir weights $\mathbf{W}$ and output weights $\mathbf{W_{out}}$ using a sigmoidal transfer function (e.g., hyperbolic tangent). ESN uses a leaky-integrated discrete-time RNN units. For such an ESN network, the update equation can be given as

$$\widetilde{\mathbf{Res}}^{t} = tanh\ (\mathbf{W_{in}}[1; \mathbf{inp}^t] + \mathbf{W}\ \mathbf{Res}^{t-1}) \tag{13}$$

$$\mathbf{Res}^{t} = (1 - \alpha_{leak})\mathbf{Res}^{t-1} + \alpha_{leak}\widetilde{\mathbf{Res}}^{t} \tag{14}$$

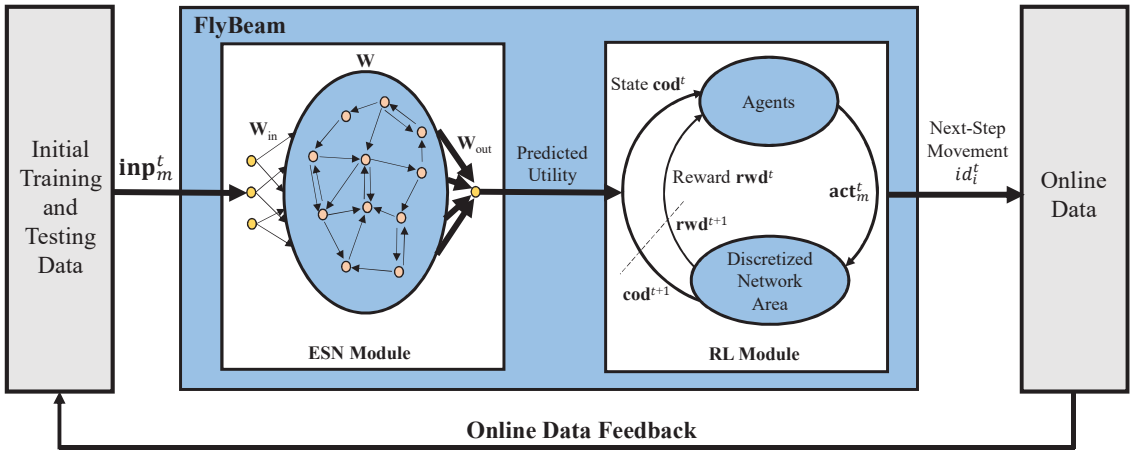


Figure 1: Diagram of the FlyBeam framework based on a combination of Echo State Learning and Reinforcement learning.

where $\mathbf{Res}^t$ is the vector of reservoir neuron activation at time t, $\widetilde{\mathbf{Res}}^t$ is the corresponding update and $\mathbf{Res}^{t-1}$ is the vector of reservoir neuron activation at time $t-1$, α_{leak} is the leaking rate, $[\cdot : \cdot]$ represents the vertical vector concatenation.

Finally, the output of the ESN $\mathbf{out}^t$ can be represented as

$$\mathbf{out}^t = \mathbf{W}_{\mathbf{out}} \, [1; \mathbf{inpt}^t; \mathbf{Res}^t] \tag{15}$$

where $[\cdot : \cdot; \cdot]$ represents the vertical vector concatenation.

Compared to traditional Neural Networks (NN), which are computationally expensive, it is incredibly simple to train ESNs, while they are still able to model the complex time-varying behaviors of dynamical systems. Furthermore, compared to traditional NNs requires more data and time to train the network which may not be desirable for UAV applications due to the fact that UAVs have limited battery life. Also, traditional NN suffers from a problem called as *vanishing/exploding gradient* in which the neurons of hidden layers do not perform as expected. In contrast, this problem is non-existent with ESN. In this work, we use ESN to approximate the utility function in (12), as described in Sec. 3, jointly considering the UAV flight, the channel estimation and beamforming strategies, as well as the effects of blockages on beamforming. Based on the ESN-approximated utility function, online RL is then adopted to guide the exploitation and exploration in favor of higher aggregate capacity. The ESN module consists of four components: *Agent*, *Input*, *Action* and *Reward Function*. In *FlyBeam*, the ESN is implemented in individual UAVs, i.e., each UAV is an *Agent*.

4.1. ESN Input Design

Let $\mathcal{T}$ denote the total network running time. In each time slot $t \in \mathcal{T}$, each UAV $m \in \mathcal{M}$ feeds an *Input* (denoted as ρ_m^t) and a candidate *Action* (denoted as ξ_m^t) to its ESN module, which will then output the expected *Reward Function* value of the UAV in the next time slot. For UAV m the input to *FlyBeam*'s ESN module in time slot t, defined as $\rho_m^t \triangleq \{\mathbf{cod}_{-m}^t\}$, comprises of the locations of all the other UAVs $\mathbf{cod}_{-m}^t = (\mathbf{cod}_{m'}^t)_{m' \in \mathcal{M}/m}$ with $\mathbf{cod}_{m'}^t$ being the coordinate vector of UAV m' in time slot t. The dimension of ρ_m^t increases quadratically[5] with the scale of the network which can slow down the training of *FlyBeam*'s ESN module and hence degrade the utility approximation accuracy in large-scale networks. To address this challenge, the network area of dimensions $L_x \times L_y \times L_z$ is divided into a number $N_x \times N_y \times N_z$ of three-dimensional rectangles, each with $\frac{L_x}{N_x}$, $\frac{L_y}{N_y}$ and $\frac{L_z}{N_z}$ for width, length and height, respectively.

[5]The input dimension is said to increase quadratically because of all the possible combinations of UAV locations.

Denote $\mathcal{N}$ as the set of the resulting rectangles. Each rectangle $n \in \mathcal{N}$ is represented using a vector $\mathbf{r}_n = (\widetilde{\mathbf{cod}}_n, \Gamma_n)$, where $\widetilde{\mathbf{cod}}_n$ is the coordinate vector of the center point of rectangle $n \in \mathcal{N}$, and $\Gamma_n = 0, 1, \cdots, N_x \times N_y \times N_z - 1$ is the index of the rectangle. Based on this policy, the input of *FlyBeam*'s ESN module can be rewritten as $\rho_m^t = (\Gamma_{m'}^t)_{m' \in \mathcal{M}/m}$, with $\Gamma_{m'}^t$ is the index of UAV m''s rectangle in time slot t.

4.2. ESN Action and Reward

Given input ρ_m^t for *FlyBeam*'s ESN module for UAV m in time slot t, UAV m makes its action decisions and observes an output of the action. To this end, UAV m chooses to move to a new rectangle in $\mathcal{N}$ except those occupied by other UAVs. The set of actions for UAV m, denoted as ξ_m^t for time slot t, can be written as

$$\xi_m^t = \{\Gamma_v | v \in \mathcal{N}/\{n^t(m'),\ m' \in \mathcal{M}/m\}\}, \tag{16}$$

where $n^t(m')$ represents the rectangle index of UAV m' in time slot t. The corresponding reward is defined as the aggregate network capacity achievable through distributed beamforming as defined by (12). Denote the resulting output capacity as C^t for time slot t.

4.3. ESN Training

We train the designed ESN module based on online collected capacity data. In the training phase, the objective of the *FlyBeam* ESN module is to learn a model with output C^t that minimizes the root-mean-square error (RMSE) between C^t (i.e., predicted sum capacity) and C_{tgt}^t (i.e., target sum capacity) defined as

$$E(C^t, C_{\text{tgt}}^t) = \frac{1}{N_{\text{out}}} \sum_{i=1}^{N_{\text{out}}} \sqrt{\frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} (C^t - C_{\text{tgt}}^t)^2}, \tag{17}$$

where N_{out} denotes the output units of the ESN, $|\cdot|$ represents the cardinality of a set and $\mathcal{T}$ denotes the number of time slots in the training phase. To this end, we measure the sum capacity based on (12) and the measured sum capacity is used as the C_{tgt}^t in (17).

Automated Meta Parameter Tuning. In order to train the ESN module and obtain an accurate prediction, the one needs to modify all the tunable parameters to achieve the optimal or at least a desirable performance. The tunable parameters of ESN includes the number of reservoir units, teacher scaling, teacher shift, input scaling and input shift denoted as N_{res}, T_s, T_{sh}, I_s, I_{sh}, respectively. However, this process of manual tuning of each parameter is time consuming and may not be accurate. Therefore, to ease the need of manual tuning of ESN parameters we design *AutoESN*. *AutoESN* uses an iterative process to update the tunable parameter. For our FlyBeam control problem, we use a sufficiently large number of reservoir units (i.e., $N_{\text{res}} >> N_{\text{inp}}$), where $N_{\text{inp}} = |\mathcal{M}|$ is the number of inputs and N_{res} is the number of reservoir units. The I_s and I_{sh} are vectors of respectively ones and zeros with a total length of the vector equal to the number of input N_{inp}. The values of I_s and I_{sh} determine the non-linearity of the reservoir response. Therefore, the optimizable parameter for our problems are T_s, T_{sh}. T_s is a multiplicative term whereas T_{sh} is an additive term. For our problem, we set the T_s to 1[6] and focus on optimizing the T_{sh}.

Denote the current value of the tunable parameter (i.e., T_{sh}) as T_{sh}^{cur} updated by a small value $\alpha \in [0, 1]$ to obtain the new parameter value T_{sh}^{new} which is then set as the current value for the next iteration as follows

$$T_{sh}^{new} \leftarrow \begin{cases} T_{sh}^{cur} + \alpha, & \text{if } \Delta_{util} < 0 \\ T_{sh}^{cur} - \alpha, & \text{if } \Delta_{util} > 0 \end{cases}, \tag{18}$$

$$T_{sh}^{cur} \leftarrow T_{sh}^{new}, \tag{19}$$

where Δ_{util} denote the difference between the predicted sum capacity C^t and target sum capacity C_{tgt}^t introduced earlier in Sec. 4.3. Since we consider for each time slot t during the training process and with the number of output units N_{out}

[6]Here, we do not optimize the T_s because it is a scaling factor and can be be set to a constant value.

set to 1, (17) can be rewritten as

$$\Delta_{util} \leftarrow C^t - C^t_{\text{tgt}}. \tag{20}$$

Here, we assume that the difference of utility is a monotonic function of the tunable parameter because at each time step of ESN training, the next step is computed based on the sum of the dot product of the weights $\mathbf{W}$ and current state of the network, $\mathbf{W_{in}}$ and scaled/shifted input vector, and feedback weight $\mathbf{W_{fb}}$ with scaled/shifted output vector. The size of input and output vectors are determined by the product of number of training samples and number of input and output units, respectively. Therefore, by increasing or decreasing the teacher shift value, the output weight $\mathbf{W_{out}}$ of the ESN also increases or decreases proportionally, thereby achieving the desired performance.

This iterative process is terminated when the following termination criteria is met.

$$\Delta_{util} \leq \Delta_{\text{th}}, \tag{21}$$

where Δ_{th} is the tolerable error between the predicted and the actual (target) utility value pre-defined before the training process. The newly tuned parameter ($T^{cur}_{sh} = T^{new}_{sh}$) can then be used for both testing, as well as, to predict the utility for any given input ρ^t_m.

4.4. ESN-Based RL Control

Based on the trained ESN-module, each UAV $m \in \mathcal{M}$ can determine its own optimal next-step location m as Γ^{t*}_m. However, this may lead to a local optimum for our swarm control problem defined in (12), which is not desirable. In this work, we use RL to guide the *exploration* and *exploitation* in the flight control of the UAVs. RL [48] has been widely used to solve complex problems that cannot be solved by conventional techniques. In this work, we consider RL algorithm with an ϵ-greedy exploration strategy [48]. As the network runs, the online collected data will be used to update the training dataset and then retrain the ESN module.

4.5. Two-Phase FlyBeam ESN Design

There are two phases of *FlyBeam*. The first one is *(i) initial training and testing phase* and the second one is *(ii) training update phase*. The former takes care of collecting the initial data samples to train the ESN model which can be used in RL control while the latter collects real time data samples which can be used to re-train the pre-trained ESN model thereby increasing the overall prediction accuracy.

Initial Training and Testing. Let $\mathcal{T}_{\text{init}}$ denote the time duration to collect the initial training data set $\text{Samp}^{\text{init}}$ expressed as

$$\text{Samp}^{\text{init}} = \begin{bmatrix} \rho^1_m & \xi^1_m & C^1 \\ \rho^2_m & \xi^2_m & C^2 \\ \vdots & \vdots & \vdots \\ \rho^{|\mathcal{T}_{\text{init}}|}_m & \xi^{|\mathcal{T}_{\text{init}}|}_m & C^{|\mathcal{T}_{\text{init}}|} \end{bmatrix}. \tag{22}$$

The collected $\text{Samp}^{\text{init}}$ data is then used to train the ESN module discussed in Sec. 4.3. The trained ESN module is further used by the RL module for flight and beamforming control introduced in Sec. 4.4.

Training Update. Let $\mathcal{T}_{\text{updt}}$ denote the periodic update time interval. As the network runs, for each time slot $t \in \mathcal{T}_{\text{updt}}$ we collect the input data and the corresponding reward as discussed in the first phase. Then, the collected data samples denoted as $\text{Samp}^{\text{updt}}$ can be expressed as

$$\text{Samp}^{\text{updt}} = \begin{bmatrix} \rho^1_m & \xi^1_m & C^1 \\ \rho^2_m & \xi^2_m & C^2 \\ \vdots & \vdots & \vdots \\ \rho^{|\mathcal{T}_{\text{updt}}|}_m & \xi^{|\mathcal{T}_{\text{updt}}|}_m & C^{|\mathcal{T}_{\text{updt}}|} \end{bmatrix}. \tag{23}$$

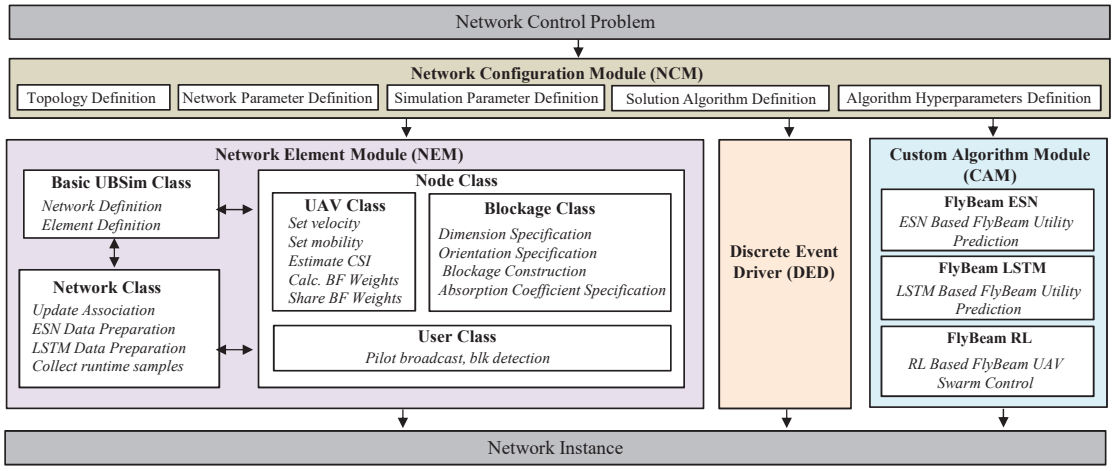


Figure 2: Diagram of the UBSim Simulator.

The collected data samples $\text{Samp}^{\text{updt}}$ are then fed back to the ESN module (as shown in Fig. 1) to update its initially trained coefficients with the newly collected data samples. This periodic update of ESN based on the collected samples during network run-time can improve the prediction accuracy of the ESN Module of *FlyBeam*.

4.6. Theoretical Proof of Convergence for ESN-Based RL Control

The FlyBeam distributed solution algorithm converges to a stationary operating point at which no UAV has an impetus to move to a new location if other UAV stays at its own location assuming mixed strategies are adopted by all the UAVs in the network.

Proof. Given the finite set of actions ξ_i^t defined in (16), denote $\Delta(\xi_i^t)$ as the set of all probability distributions over the elements of ξ_i^t and $\mathbb{P}_i \in \Delta(\xi_i^t)$ as the probability distribution used by UAV $i \in \mathcal{M}$ to select an action from its action set ξ_i^t . Then, the mixed strategy profile for UAV i, denoted as $\mathbb{P}_i^* \in \Delta(\xi_i^t), \forall\ i \in \mathcal{M}$, can be given as $\mathbb{P}_i^* = [\mathbb{P}_i^*(\Gamma_1), \cdots, \mathbb{P}_i^*(\Gamma_{|\mathcal{N}|})]$, where $|\mathcal{N}|$ being the cardinality of the set of rectangles defined in Section 4.1. The flight control and collaborative beamforming can then be reformulated as a non-cooperative game and the theorem can be proved by showing that this game converges to a mixed Nash Equilibrium (MNE) with mixed strategy probability [49], that is, we need to show the following condition holds for a mixed strategy profile $\mathbb{P}_i^* = [\mathbb{P}_i^*(\Gamma_1), \cdots, \mathbb{P}_i^*(\Gamma_{|\mathcal{N}|})] = (\mathbb{P}_i^*, \mathbb{P}_{-i}^*)$, i.e., $\tilde{u}_i(\mathbb{P}_i^*, \mathbb{P}_{-i}^*) \geq \tilde{u}_i(\mathbb{P}_i, \mathbb{P}_{-i}^*)$.

Recall that ϵ-greedy exploration strategy is adopted in FlyBeam to ensure that the probability of choosing an action ξ_i^t is always greater than 0. Then, the probability of UAV $i \in \mathcal{M}$ choosing an action ξ_i^t can be given as

$$Prob(\xi_i^t) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\xi_i^t|}; & \arg\max_{\xi_i^t} C(\xi_i^t) \\ \frac{\epsilon}{|\xi_i^t|}; & \text{a random action} \end{cases} \tag{24}$$

Let $\widetilde{\xi}_i^{t+1,*}$ denote the action that results in optimal reward given the optimal mixed strategy $(\mathbb{P}_i^*, \mathbb{P}_{-i}^*)$. Then the

utility function of UAV i can be given as

$$\tilde{u}_i(\mathbb{P}^*_i, \mathbb{P}^*_{-i}) - \tilde{u}_i(\mathbb{P}_i, \mathbb{P}^*_{-i}) = \sum_{\widetilde{\xi}_i^{t+1} \in \xi_i^t} [\mathbb{P}^*_{i,\widetilde{\xi}_i^{t+1}} \sum_{\widetilde{\xi}_{-i}^{t+1} \in \xi_{-i}^t} u_i(\xi_i^{t+1}, \xi_{-i}^{t+1}) \mathbb{P}^*_{i,\xi_{-i}^{t+1}} - \mathbb{P}_{i,\widetilde{\xi}_i^{t+1}} \sum_{\widetilde{\xi}_{-i}^{t+1} \in \xi_{-i}^t} u_i(\xi_i^{t+1}, \xi_{-i}^{t+1}) \mathbb{P}^*_{i,\xi_{-i}^{t+1}}] \tag{25}$$

$$= \sum_{\widetilde{\xi}_i^{t+1} \in \xi_i^t} \mathbb{E}[u_i(\widetilde{\xi}_i^{t+1})] \{\mathbb{P}^*_{i,\widetilde{\xi}_i^{t+1}} - \mathbb{P}_{i,\widetilde{\xi}_i^{t+1}}\} \tag{26}$$

$$= (1-\epsilon) \left(\mathbb{E}[u_i(\widetilde{\xi}_i^{t+1,*})] - \mathbb{E}[u_i(\widetilde{\xi}_i^{t+1})] \right) \tag{27}$$

where (27) is obtained from (24). We can then conclude that $\mathbb{E}[u_i(\widetilde{\xi}_i^{t+1,*})] - \mathbb{E}[u_i(\widetilde{\xi}_i^{t+1})] \geq 0$ based on the fact that in *FlyBeam* the optimal action $\xi_i^{t+1,*}$ results in optimal $\mathbb{E}[u_i(\widetilde{\xi}_i^{t+1,*})]$ and the value of $\mathbb{E}[u_i(\widetilde{\xi}_i^{t+1})$ cannot exceed the optimal value and therefore the difference is always greater than or equal to 0. This completes the proof. ■

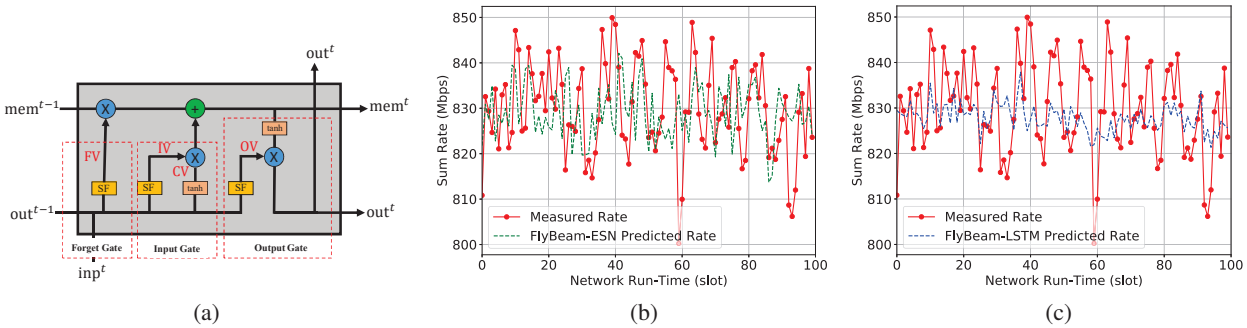
Notation	Physical Meaning	Value
$L_x \times L_y \times L_z$	Network area	200 ×200 × 200
$N_x \times N_y \times N_z$	3-D rectangle dimensions	20 ×20 × 20
$L_x/N_x \times L_y/N_y \times L_z/N_z$.	Number of grids	10 ×10 × 10
f	Frequency	2.1 GHz
B	Bandwidth	40 MHz
P_{mu}	Transmit Power	1 W
$\beta_{mu}(f)$	Path-loss exponents (LOS, NLOS)	2, 4
$\mathcal{T}$	Initial Training time slots	3000
$\mathcal{T}_{\text{updt}}$	Training update time slots	1000
$\mathcal{T}_{\text{coh}}$	Channel Coherence Time	100 ms

Table 2: Key Simulator Configuration Variables.

5. Performance Evaluation

We consider a swarm UAV network with area of $200 \times 200 \times 200$ m^3. The frequency and bandwidth are set to 2.1 GHz and 40 MHz, respectively. The transmission power of each UAV is set to 1 W. The LOS and NLOS path-loss exponents are considered to be 2 and 4, respectively. The duration of each time slot is set to 100 ms (equal to coherence time). The users, UAVs and blockages are uniformly distributed in the network area. Two beamforming schemes are considered: *Zero Forcing (ZF)* and *Maximum Ratio Transmission (MRT)* [46]. The corresponding *FlyBeam* schemes are referred to as *FlyBeam-ZF* and *FlyBeam-MRT*, respectively. We set $\mathcal{T}$ and $\mathcal{T}_{\text{updt}}$ as 3000 and 1000 time slots for initial training and training update phases, respectively. The simulations have been conducted over *UBSim*, a newly developed event-driven universal broadband simulator for integrated aerial and ground wireless networking by modifying the earlier version of UBSim (and SimBAG) designed as a port of our prior research [3], [4], to include additional features such as channel estimation, channel state information measurement, among others.. In addition to the beamforming schemes discussed in this paper, UBSim also supports other network control problems that deals with multiple spectrums including support for 6 GHz band, ground and flying base stations, integrated access and backhaul links, beam alignment for directional communications. In addition to the above mentioned features, we implemented a benchmark scheme based on UBSim for performance comparison. The major notations and their values are summarized in Table. 2.

UBSim Simulator Design. The overall architecture of UBSim simulator is shown in Fig. 2. UBSim is a Python based event-driven simulator for universal broadband integrated aerial-ground wireless networks. UBSim compries of four major modules, namely *Network Configuration Module (NCM), Network Element Module (NEM), Discrete Event Driver (DED), and Custom Algorithm Module (CAM). The NCM module allows users to define the network topology, define the network parameters such as the number of UAVs, the number of users; simulation parameters such as*



(a) (b) (c)

Figure 3: (a) Single-Cell LSTM Architecture; Comparison of Prediction Accuracy (b) FlyBeam-ESN ; (c) FlyBeam-LSTM.

simulation time, iteration count, among others, and finally set the hyperparameters required for different solution algorithms. The NEM module defines the classes for all the network elements, such as network, UAVs, blockages, users, links, interference, among others. All these classes follow a hierarchical architecture with the general network element class at the highest level. This class defines the basic network element attributes and operations such as registering an element in the network, specifying the parent and children elements of an element. The discrete network simulation is orchestrated by the DED, which is designed based on the open-source library SimPy [50]. Finally, the CAM module hosts the custom-designed network control algorithms discussed in Sec. 4, such as FlyBeam ESN, FlyBeam LSTM as well as FlyBeam RL.

5.1. Benchmark: Lower Bound Without Beamforming

In this benchmark scheme, no collaborative beamforming is considered, and the UAVs access the channel in a frequency division multiple access (FDMA) manner and further serve their users based on time division multiple access (TDMA). Each UAV maximizes the aggregate capacity of the users it serves based on Sequential Least Squares Programming (SLSQP) [51], to dynamically allocate their transmission power among different sub-channels to avoid mutual interference.

$$\begin{aligned} &\textit{Given}: && \mathcal{S}, \mathcal{M}, \mathcal{U}, \mathcal{Q}, B, \mathbf{cod}_u, \mathbf{cod}_m, \forall u \in \mathcal{U}, \forall m \in \mathcal{M} \\ &\textit{Maximize}: && \sum_{m\in\mathcal{M}} \sum_{q\in\mathcal{Q}} \sum_{u\in\mathcal{U}_m} C_{mqu} \\ &\textit{Subject to}: && \sum_{q\in\mathcal{Q}} \sum_{u\in\mathcal{U}_m} P_{mqu} \le P_{\max}, \ \forall m \in \mathcal{M} \end{aligned} \tag{28}$$

$$C_{mqu} = B_q \log_2\Big(1 + \frac{P_{mqu} L_{mqu}}{\sum\limits_{m'\in\mathcal{M}/m} \sum\limits_{u'\in\mathcal{U}_{m'}} P_{m'qu'} L_{m'qu'} + N_{mqu}}\Big) \tag{29}$$

where $\mathcal{Q}$ represents the set of subchannels and B_q is the bandwidth of subchannel $q \in \mathcal{Q}$; P_{mqu} is the transmission power UAV m allocates to user $u \in \mathcal{U}_m$ on subchannel $q \in \mathcal{Q}$; here, $\mathcal{U}_m$ denotes the set of users served by UAV m, which is determined based on the *shortest-distance association policy*, i.e., each user associates to the nearest UAV; N_{mqu} is the noise power at user u on subchannel $q \in \mathcal{Q}$.

5.2. Effectiveness Validation of ESN Learning

We first validate the effectiveness of ESN learning. To this end, we compare ESN-based training to that based on Long Short-Term Memory (LSTM) [16–20] To this end, we design an LSTM-based utility prediction algorithm that can be used to solve the FlyBeam control problem (12). LSTM networks are a special kind of Recurrent Neural Network (RNN), capable of learning long-term dependencies. A typical LSTM network is comprised of different memory blocks called *cells* as shown in Fig. 3(a). The main function of these blocks is to remember and manipulate

the information to the memory (mem^{t-1}). This is achieved through three major mechanisms, called *gates*. In general, a LSTM cell consists of three gates, namely *forget gate, input gate* and *output gate* (red dashed box in Fig. 3(a)).

Forget Gate. The forget gate is responsible for removing information that is of less importance from the cell state using a sigmoid function (SF) so that the performance of the LSTM network is optimized. To this end, the forget gate outputs a value between 0 and 1 for each cell memory state (mem^{t-1}) by comparing output (out^{t-1}) and input (inp^{t}). If the value is closer to 1, then the information is retained and if it is closer to 0, then the information can be discarded. Forget Vector (FV) can be represented as

$$FV = SF(W_F \cdot [\text{out}^{t-1}, \text{inp}^{t}] + \text{B}_\text{F}^{t}), \tag{30}$$

where, W_F denotes the weight matrix between the forget and the input gates, B_F^t denotes the associated bias.

Input Gate. The input gate is responsible for adding information to the cell state using sigmoid and hyperbolic tangent (tanh) functions. The tanh functions creates a vector of new candidate values. Then, input vector (IV) and candidate values vector (CV) can be expressed as

$$IV = SF(W_I \cdot [\text{out}^{t-1}, \text{inp}^{t}] + \text{B}_\text{I}^{t}), \tag{31}$$

$$CV = TH(W_C \cdot [\text{out}^{t-1}, \text{inp}^{t}] + \text{B}_\text{C}^{t}), \tag{32}$$

where W_I and W_C denotes the weight matrix of sigmoid function between the input and output gates and the associated biases are denoted by B_I^t and B_C^t, respectively.

Next step is to store the information to the memory cell (mem^{t}). This is done by multiplying the previous cell state (mem^{t-1}) with forget vector (FV)

$$\text{mem}^{t} = \text{FV} * \text{mem}^{t-1} + \text{IV} * \text{CV}, \tag{33}$$

If the obtained result is 0, then the values will get dropped in the cell state mem^{t}

Output Gate The output gate is responsible for determining the output by selecting useful information from the current cell state leveraging the capabilities of tanh function and a filter designed based on sigmoidal function. This output is then sent to the hidden state of the next cell. The output vector (OV) and the LSTM output mem^{t} can be expressed as

$$OV = SF(W_O \cdot [\text{out}^{t-1}, \text{inp}^{t}] + \text{B}_\text{O}^{t}), \tag{34}$$

$$\text{out}^{t} = \text{OV} * \tanh(\text{mem}^{t}), \tag{35}$$

where W_O denotes the weight matrix of output vector and the associated biases is denoted by B_O^t.

We first compare the prediction accuracy of *FlyBeam-ESN* and *FlyBeam-LSTM*. The LSTM-based utility prediction model (*FlyBeam-LSTM*) is designed following the principles of the three gates and leveraging some key open-source libraries available for LSTM such as "adam" optimizer [52], "mean squared error" loss function [53] and MinMaxScaler [54], among others. The results are reported in Fig. 3. In this experiment, we consider 1000 time slots for the initial training data samples and use the first 900 data samples for training and the remaining for testing. For LSTM, we use the same dataset with epoch counter set to 50 iterations. As we can see from the Figs. 3 (b) and (c), *FlyBeam-ESN* is able to predict the utility at least with 75% accuracy while it is less than 40% for *FlyBeam-LSTM*. This verifies the effectiveness of ESN-based utility prediction with limited training samples, and we will focus on ESN-based FlyBeam in the following performance evaluation.

Next, we analyse the effectiveness of the designed autotuning framework (*AutoESN*). In this experiment, we use the same training data as that used to train the model used in Fig. 3(b). As discussed in Sec. 4.3, in general ESN has a variety of parameters such as number of reservoirs, input shift, input scaling, teacher shift, teacher scaling as well as the activation functions. For the FlyBeam problem, we mainly focus on the *teacher shift* which is an additive term applied to the target signal. By optimizing the teacher shift (ts) value, the loss function Δ_{util} defined in (20) can be reduced. The number of reservoirs is set to 100, input shift vector and input scaling vector are set to 0 and 1, respectively. The value of teacher shift is varied from -1 to +1 in steps of 0.2 and the value of $alpha$ (refer (18)) is set

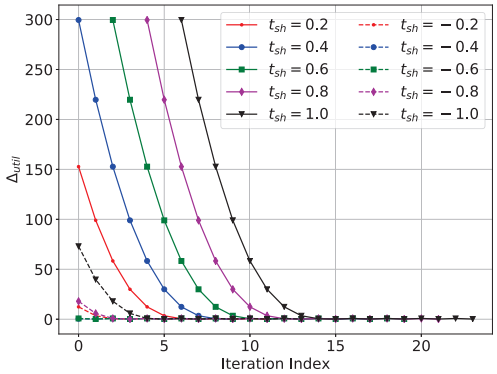
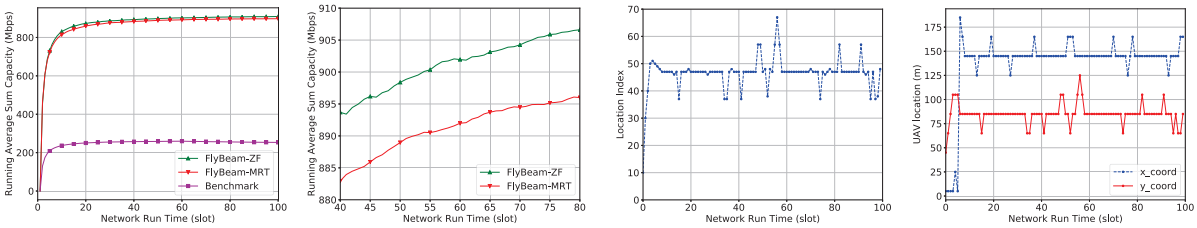


Figure 4: Convergence of loss value function Δ_{util} with different teacher shift values.



(a) (b) (c) (d)

Figure 5: (a) Running average sum capacity (b) Running average sum capacity for time slots 40 to 80 ; (c) Location indices of one UAV and (d) X- and Y- coordinate of one UAV.

as 0.1[7]. The resulting loss function (Δ_{util}) is reported in Fig. 4. It can be seen that irrespective of the starting value of the "ts", *AutoESN* is able to converge to a value of −0.6 which is the optimal teacher shift value for the considered training dataset. The newly optimized teacher shift value can then be used in the testing phase.

5.3. *Performance Analysis*

In the first experiment we analyze the running average sum capacity of the network as the UAVs move to different location based on the FlyBeam ESN-RL algorithm for a network with 3 UAVs collaboratively serving 10 users. We compare FlyBeam (*FlyBeam-ZF and FlyBeam MRT*) with the benchmark scheme and the results are reported in Fig. 5. In Fig. 5(a), we show the running average sum capacity of the 3-UAV network and Fig. 5(b) shows the running average sum capacity of the network between the time slots 40 to 80 for better visualisation. We consider UAV 1 as an example and plot its location index in Fig. 5(c) and the corresponding x- and y- coordinates in Fig. 5(d). It is important to note here that the fluctuations in the location index (Fig. 5(c)) is not random, but based on the movement of the UAV in upward or downward direction. For example, at time slot 14, the location index is *47* and at time slot 15, the location index is *37*, this is due to the fact that the UAV took an action to move down. Recall from Table 2 that there are 10 grids in each row. This downward movement can be further verified from Fig. 5(d) where the y-coordinate value changes from 85m to 65m. This is because, the distance between two grid points in 20m (refer Table 2). The similar trend can be observed at time slots 54 and 55 as the UAV moves up and at time slots 56 and 57, when the UAV

[7]Please note that the value of α can be set to a value less than 0.1 for applications that require even finer tuning. For FlyBeam problem, with $\alpha = 0.01$ and $\alpha = 0.001$, the value of teacher shift (ts) converges to −0.56 and −0.551 and takes 34 and 252 iterations, respectively.

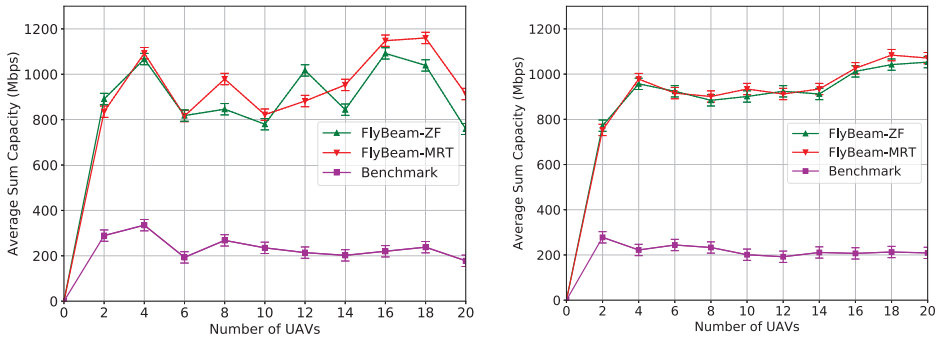


Figure 6: Average sum capacity with different number of UAVs using (a) Sequential Channel Estimation ; (b) Simultaneous Channel Estimation.

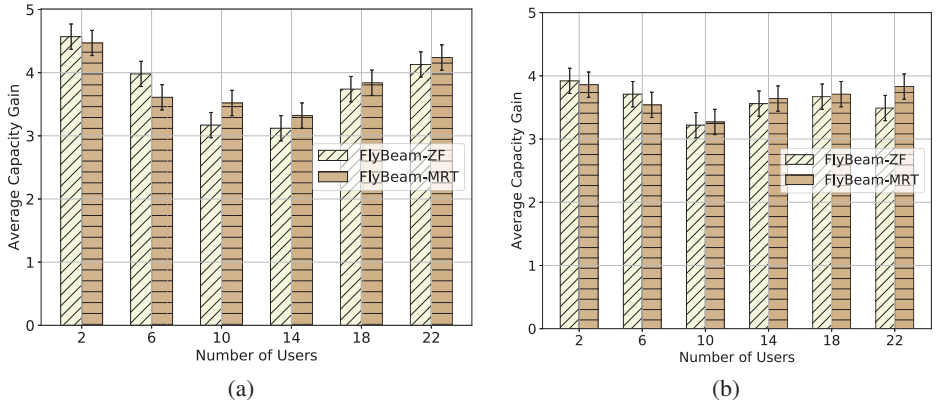


Figure 7: Capacity gain with different number of users using (a) Sequential Channel Estimation ; (b) Simultaneous Channel Estimation.

moves down. Furthermore, at time slots 18 and 19 as the UAV moves right or left, the change in x-coordinate can be seen in Fig. 5(d).

Next, we evaluate the capacity performance of *FlyBeam*. The number of UAVs is varied from 2 to 30 in step of 2 and the number of ground users is set to 10. The results are reported in Fig. 6(a) and (b) for sequential and simultaneous channel estimation, respectively. It can be seen from Fig. 6(a) that significant capacity gain can be achieved by *FlyBeam*. For example, average capacities of 915.20 Mbps and 960.12 Mbps can be achieved with *FlyBeam-ZF* and *FlyBeam-MRT*, respectively, which are 3.56 and 3.73 times higher than that achievable by Benchmark Scheme 2 (257 Mbps). Similar results can also be observed in Fig. 6(b) for simultaneous channel estimation.

The effectiveness of distributed beamforming in swarm UAV networks is further verified in Fig. 7 in terms of average capacity gain with 10 UAVs and the number of users varying from 2 to 22 in steps of 4. We can see that with sequential channel estimation both *FlyBeam-ZF* and *FlyBeam-MRT* are able to achieve a capacity gain between 312% and 457% with an average of 380% as shown in Fig. 7(a). This observation is further verified with simultaneous channel estimation as shown in Fig. 7(b) where we can see that *FlyBeam-ZF* and *FlyBeam-MRT* are able to achieve a capacity gain between 322% and 392% with an average of 360%. It is worth mentioning here that For the performance evaluation, we consider random distribution of blockages, altitude of blockages, users and UAVs in the network at the start for different cases studied. Therefore, there are fluctuations in the obtained results in Fig. 7. However, it is important to note here that, the key takeaway from this analysis is the performance gain of FlyBeam-MRT and

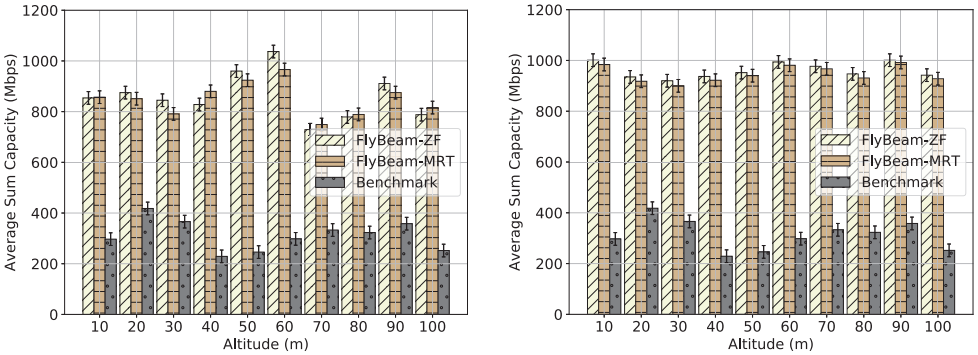


Figure 8: Average sum capacity with varying flight altitude using (a) Sequential Channel Estimation ; (b) Simultaneous Channel Estimation.

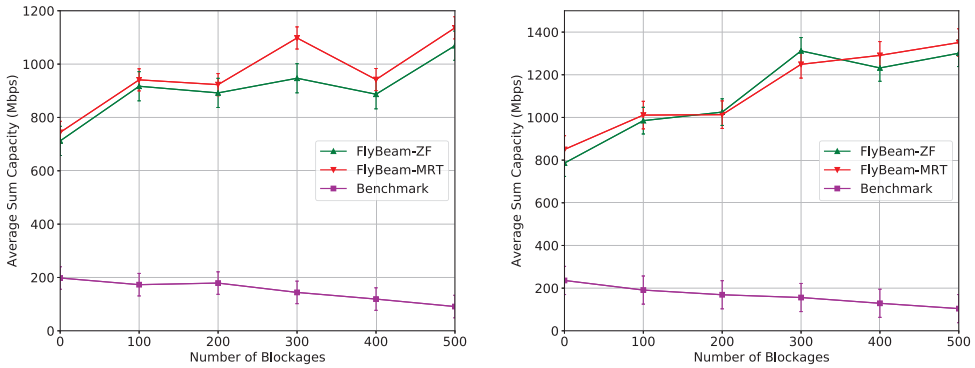
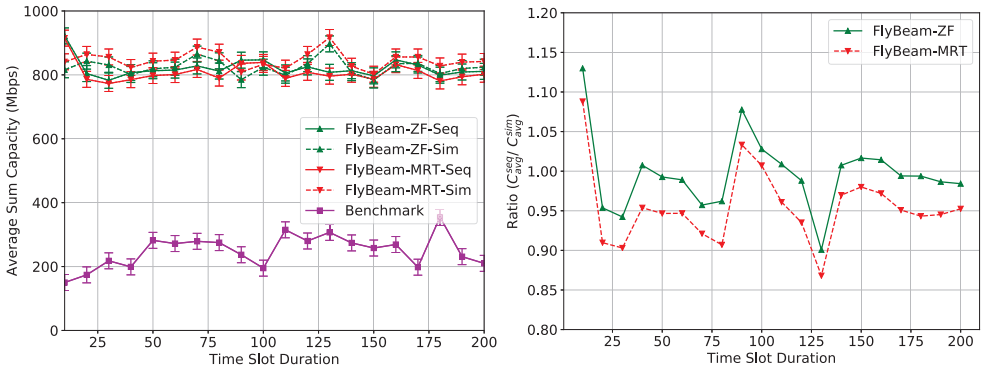


Figure 9: Average sum capacity with varying number of blockages using (a) sequential channel estimation and (b) simultaneous channel estimation.

FlyBeam-ZF compared to the benchmark scheme.

In this experiment, we further investigate the effects of the UAVs' flight altitude on the beamforming gain, considering 10 ground users and 10 UAVs. The flight altitude of the UAVs is varied from 10 m to 100 m in steps of 10 m. The results are plotted in Fig. 8. Similar to that in Figs. 6 and 7, significant capacity gains can be achieved by *FlyBeam* in all the tested cases. It can also be found that the sum capacity achievable by *FlyBeam* is only slightly affected by the UAV flight altitude. For example, with sequential channel estimation as shown in Fig. 8(a) an average sum capacity of 854 Mbps can be achieved by *FlyBeam-ZF* with a flight altitude of 10 m, which is 816 Mbps for an altitude of 100 m. Similar results can also be achieved by the *FlyBeam-MRT*. This performance can also be observed with simultaneous channel estimation as shown in Fig. 8(b). This is expected because as discussed in Section 1, while flying lower can reduce the propagation distance to the ground users, the transmissions may experience higher attenuation because of a higher probability of being blocked.

The effects of blockages on the beamforming gain is further studied in Fig. 9. The number of blockages is varied from 0 to 500 in steps of 100. The minimum and maximum dimensions of the blockages are set to 2 m and 8 m for length and width, respectively. The height of the blockages is fixed to 15 m, and the UAV flight altitude is set to 30 m. It can be found in Fig. 9(a) that with sequential channel estimation *FlyBeam-ZF* and *FlyBeam-MRT* can yield respectively an average capacity of 1106.9 Mbps and 1127.5 Mbps, which are 6.74 and 6.87 times larger than Benchmark Scheme 2. The sum capacity achievable with Benchmark Scheme 2 decreases monotonically with



(a) (b)

Figure 10: (a) Average sum capacity with different slot duration; (b) Ratio of average sum capacity with Sequential Channel Estimation to average sum capacity with Simultaneous Channel Estimation.

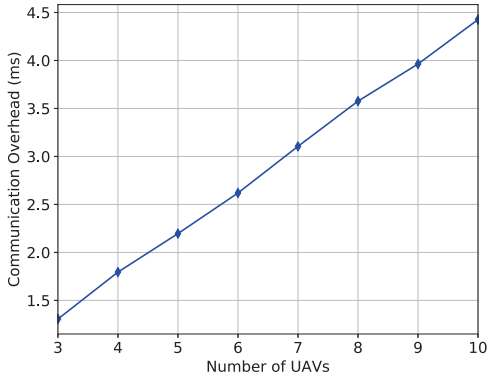


Figure 11: Communication overhead of *FlyBeam*.

the number of blockages because of larger signal attenuation with more blockages. A surprising observation is that a *higher rather than lower* sum capacity can be achieved by *FlyBeam* with more blockages in the network. For example, an average sum capacity of 786 Mbps can be achieved by the *FlyBeam-ZF* with no blockages, which are 1025 Mbps and 1301 Mbps with 200 and 500 blockages, respectively. This is primarily because denser blockages introduce more diversity in the wireless channels, and hence higher spatial diversity gain can be achieved through beamforming. The similar performance can be seen with simultaneous channel estimation as well as shown in Fig. 9(b).

We further study the effect of varying time slot duration on the average sum capacity and the results are reported in Fig. 10(a). The time slot duration was varied from 10 to 200 in steps of 10. We consider FlyBeam-ZF and FlyBeam-MRT schemes with sequential (*FlyBeam-ZF-Seq, FlyBeam-MRT-Seq*) as well as simultaneous (*FlyBeam-ZF-Sim, FlyBeam-MRT-Sim*) channel estimation methods. The corresponding ratio of average sum capacity with sequential channel estimation to that with simultaneous channel estimation is shown in Fig. 10(b). As expected, the network performance with collaborative beamforming outperforms the distributed benchmark scheme with an average sum capacity of 825 Mbps, which is 3.31 times larger than that of the benchmark scheme (249 Mbps). Moreover, Fig. 10(b) shows that it is more desirable for UAVs to collaborate in distributed beamforming based on zero-force precoding and simultaneous channel estimation. It is worth mentioning here that in the performance evaluation we compared the average capacity gain and average sum capacity for FlyBeam-MRT and FlyBeam-ZF compared to the benchmark scheme. Here for FlyBeam-MRT and FlyBeam-ZF we considered collaborative beamforming where multiple UAVs can serve a single user. However, in the benchmark scheme we consider FDMA for UAVs channel access and TDMA to serve the users. Due to this, the average capacity (or gain) achieved is lower compared to the

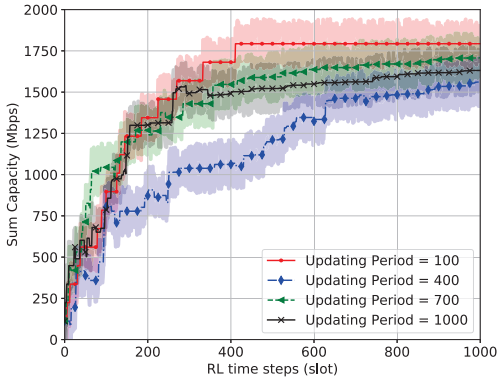


Figure 12: Convergence of *FlyBeam* with different updating periods.

collaborative beamforming achieved by FlyBeam-MRT and FlyBeam-ZF. Furthermore, at the start of the simulation, all the UAVs and users are randomly placed. At this point, the link between UAVs and users may be LOS dominant or NLOS dominant. Therefore, by allowing the UAVs to explore the network area it can determine the sweet-spot where the best network performance is achieved.

5.4. ***Complexity and Convergence***

We study the computational complexity and communication overhead of *FlyBeam* control in Fig. 11. The experiments are conducted on a workstation with Intel(R) Core(TM) *i*7 − 10510U CPU @ 1.80 GHz 2.30 GHz, memory of 16.0 GB, and 64-bit Windows Operating System. We fix the number of users as 10 and vary the number of UAVs from 3 to 10 at a step of 1. Figure 11(b) plots the corresponding communication overhead, i.e., the time taken for CSI and beamforming weights sharing. We can see with 3 UAVs, the time taken is 1.31 ms which yields 98.69 ms for the actual communication which is 98% of the coherence time. Furthermore, we can see that with 10 UAVs, the time taken is 4.43 ms yielding 95.57% of coherence time for the communication. *This verifies that with our proposed algorithm has very little communication overhead in comparison to the available time for the communication.*

We further study the effects of periodic updating of the ESN training on the network's sum capacity, taking *FlyBeam-MRT* as an example. The results are plotted in Fig. 12, where each curve is obtained by averaging over 20 simulations. The updating period is varied from 100 to 1000 time slots at steps of 300 time slots. It can be seen that the convergence speed is significantly affected by the updating period. In most of the test cases, *FlyBeam* converges faster with shorter updating periods. For example, *FlyBeam* converges in around 400 time slots with an updating period of 100 time slots, which is twice as fast as that with updating periods of 700 and 1000 time slots. However, interestingly, we found that there is no monotonic mapping between the convergence speed and the updating period. This can be seen when the updating period is set to 400 time slots. In future work, we will investigate the optimal updating frequency for online RL by considering the involved interaction between ESN-based utility function approximation and RL-based state exploration.

6. Conclusions

In this paper we proposed a high-data-rate swarm UAV network with distributed beamforming capabilities by taking into account different factors that affect the beamforming gain. We designed *FlyBeam* to jointly control the flight and beamforming in swarm UAV networks, based on a combination of ESN learning and RL. We considered two beamforming schemes (*ZF and MRT*) as well as two channel estimation schemes (*sequential and simultaneous*). We verified the performance of *FlyBeam* by considering two benchmark solution schemes based on traditional optimization and LSTM-based utility prediction. The performance evaluation was conducted over a newly developed simulator called *UBSim*. Simulation results indicate that i) up to 450% capacity gain can be achieved by enabling distributed beamforming in swarm UAV networks, and ii) with distributed beamforming, *higher (rather than lower)*

network capacity can be achieved with denser blockages. In the future, we will further verify the performance of FlyBeam on software-defined radio based integrated aerial-ground network experimentation facilities being developed at University at Buffalo [55].

References

[1] S. K. Moorthy, Z. Guan, S. Pudlewski, E. S. Bentley, FlyBeam: Echo State Learning for Joint Flight and Beamforming Control in Wireless UAV Networks, in: Proc. of IEEE International Conference on Communications (ICC), Virtual/Montreal, Canada, 2021.

[2] V. Sharma, R. Kumar, A Cooperative Network Framework for Multi-UAV Guided Ground Ad hoc Networks, Journal of Intelligent Robotic Systems 77 (2015) 629–652.

[3] S. K. Moorthy, Z. Guan, Beam Learning in MmWave/THz-band Drone Networks Under In-Flight Mobility Uncertainties, IEEE Transactions on Mobile Computing 21 (6) (2022) 1945–1957.

[4] S. K. Moorthy, Z. Guan, FlyTera: Echo State Learning for Joint Access and Flight Control in THz-enabled Drone Networks, in: IEEE International Conference on Sensing, Communication, and Networking (SECON), 2020.

[5] K. G. Panda, S. Das, D. Sen, W. Arif, Design and Deployment of UAV-Aided Post-Disaster Emergency Network, IEEE Access 7 (2019) 102985–102999.

[6] M. M. Azari, G. Geraci, A. Garcia-Rodriguez, S. Pollin, Cellular UAV-to-UAV Communications, in: Proc. of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Istanbul, Turkey, 2019.

[7] Q. Wu, R. Zhang, Common Throughput Maximization in UAV-Enabled OFDMA Systems With Delay Consideration, IEEE Transactions on Communications 66 (12) (2018) 6614–6627.

[8] Q. Wu, Y. Zeng, Joint Trajectory and Communication Design for Multi-UAV Enabled Wireless Networks, IEEE Transactions on Wireless Communications 17 (3) (2018) 3417–3442.

[9] L. Bertizzolo, S. D'Oro, L. Ferranti, L. Bonati, E. Demirors, Z. Guan, T. Melodia, S. Pudlewski, SwarmControl: An Automated Distributed Control Framework for Self-Optimizing Drone Networks, in: IEEE Conference on Computer Communications (INFOCOM), 2020.

[10] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, M. Debbah, A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems, IEEE Commununications Surveys & Tutorials 21 (3) (2019) 2334–2360.

[11] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, J. Yuan, Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges, IEEE Communications Surveys & Tutorials 21 (4) (2019) 2109–2121.

[12] H. Peng, C. Chen, C.-C. Lai, L.-C. Wang, Z. Han, A Predictive On-Demand Placement of UAV Base Stations Using Echo State Network, in: Proc of IEEE/CIC International Conference on Communications in China (ICCC), Changchun, China, 2019.

[13] X. Liu, Y. Liu, Y. Chen, L. Wang, Z. Lu, Machine Learning Aided Trajectory Design and Power Control of Multi-UAV, in: Proc of IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, 2019.

[14] M. Chen, W. Saad, C. Yin, Echo State Learning for Wireless Virtual Reality Resource Allocation in UAV-Enabled LTE-U Networks, in: Proc of IEEE International Conference on Communications (ICC), Kansas City, MO, 2018.

[15] X. Liu, Y. Liu, Y. Chen, L. Hanzo, Trajectory Design and Power Control for Multi-UAV Assisted Wireless Networks: A Machine Learning Approach, IEEE Transactions on Vehicular Technology 68 (8).

[16] K. Li, W. Ni, F. Dressler, LSTM-characterized Deep Reinforcement Learning for Continuous Flight Control and Resource Allocation in UAV-assisted Sensor Network, IEEE Internet of Things Journal.

[17] Y. Lin, M. Wang, X. Zhou, G. Ding, S. Mao, Dynamic Spectrum Interaction of UAV Flight Formation Communication With Priority: A Deep Reinforcement Learning Approach, IEEE Transactions on Cognitive Communications and Networking 6 (3) (2020) 892 – 903.

[18] C. Liu, W. Yuan, Z. Wei, X. Liu, D. W. K. Ng, Location-Aware Predictive Beamforming for UAV Communications: A Deep Learning Approach, IEEE Wireless Communications Letters 10 (3) (2021) 668 – 672.

[19] H. Yao, Y. Liu, X. Zhang, Developing deep LSTM model for real-time path planning in unknown environments, in: Proc. of IEEE International Conference on Dependable Systems and Their Applications (DSA), Xi'an, China, 2020.

[20] M. M. Azari, G. Geraci, A. Garcia-Rodriguez, S. Pollin, Cellular UAV-to-UAV Communications, in: Proc. of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Istanbul, Turkey, 2019.

[21] H. Huang, Y. Yang, H. Wang, Z. Ding, H. Sari, F. Adachi, Deep Reinforcement Learning for UAV Navigation Through Massive MIMO Technique, IEEE Transactions on Vehicular Technology 68 (1) (2020) 1117–1121.

[22] J. Cui, Y. Liu, A. Nallanathan, Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks, IEEE Transactions on Wireless Communications 19 (2) (2020) 729–743.

[23] U. Challita, W. Saad, C. Bettstetter, Interference Management for Cellular-Connected UAVs: A Deep Reinforcement Learning Approach, IEEE Transactions on Wireless Communications 18 (4) (2019) 2125–2140.

[24] L. Wang, P. Huang, K. Wang, G. Zhang, L. Zhang, N. Aslam, K. Yang, RL-Based User Association and Resource Allocation for Multi-UAV enabled MEC, in: Proc. of IEEE International Wireless Communications Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019.

[25] S. Mohanti, C. Bocanegra, J. Meyer, G. Secinti, M. Diddi, H. Singh, K. Chowdhury, AirBeam: Experimental Demonstration of Distributed Beamforming by a Swarm of UAVs, in: Proc. of IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Monterey, CA, USA, Nov. 2019.

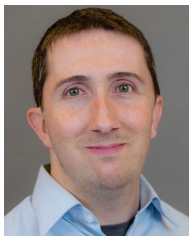
[26] Y. Lu, J. Fang, Z. Guo, J. A. Zhang, Distributed Transmit Beamforming for UAV to Base Communications, China Communications 16 (1) (2019) 15–25.

[27] S. Yeh, J. Chamberland, G. H. Huff, An Investigation of Geolocation-Aware Beamforming Algorithms for Swarming UAVs, in: Proc. of IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting, San Diego, CA, USA, 2017.

[28] P. Dinh, T. M. Nguyen, C. Assi, W. Ajib, Joint Beamforming and Location Optimization for Cooperative Content-Aware UAVs, in: Proc. of IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 2019.

[29] W. Yuan, C. Liu, F. Liu, S. Li, D. W. K. Ng, Learning-based Predictive Beamforming for UAV Communications with Jittering, IEEE Wireless Communications Letters 9 (11).
[30] W. Miao, C. Luo, G. Min, L. Wu, T. Zhao, Y. Mi, Position-Based Beamforming Design for UAV Communications in LTE Networks, in: Proc. of IEEE International Conference on Communications (ICC), Shanghai, China, 2019.
[31] I. Kocaman, Distributed beamforming in a swarm UAV network, Calhoun: The NPS Institutional Archive DSpace Repository.
URL https://calhoun.nps.edu/handle/10945/4215
[32] Z. Guan, N. Cen, T. Melodia, S. Pudlewski, Joint Power, Association and Flight Control for Massive-MIMO Self-Organizing Flying Drones, IEEE/ACM Transactions on Networking 28 (4) (2020) 1491–1505.
[33] D. C. Araújo, T. Maksymyuk, A. L. F. de Almeida, T. Maciel, J. C. M. Mota, M. Jo, Massive MIMO: Survey and Future Research Topics, IET Communications 10 (15) (2016) 1938–1946.
[34] P. Zhang and J. Chen and X. Yang and N. Ma and Z. Zhang, Recent Research on Massive MIMO Propagation Channels: A Survey, IEEE Communications Magazine 56 (12) (2018) 22–29.
[35] M. Y. Arafat, S. Moh, Localization and Clustering Based on Swarm Intelligence in UAV Networks for Emergency Communications, IEEE Internet of Things Journal 6 (5) (2019) 8958–8976.
[36] S. Bhandari, X. Wang, R. Lee, Mobility and Location-Aware Stable Clustering Scheme for UAV Networks, IEEE Access 8 (2020) 106364–106372.
[37] K. Ntontin, C. Verikoukis, Toward the Performance Enhancement of Microwave Cellular Networks Through THz Links, IEEE Transactions on Vehicular Technology 66 (7) (2017) 5635–5646.
[38] T. Bai, R. Vaze, R. W. Heath, Analysis of Blockage Effects on Urban Cellular Networks, IEEE Transactions on Wireless Communications 13 (9) (2014) 5070–5083.
[39] Spatial channel model for multiple input multiple output (MIMO) simulations (3GPP TR 25.996 version 11.0.0 release 11).
URL https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html
[40] J. Dang, C. Li, Y. Meng, Research on Least Square Channel Estimation Algorithm Based on Phase Compensation, in: Proc. of IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 2016.
[41] K. Appaiah, A. Ashikhmin, T. L. Marzetta, Pilot Contamination Reduction in Multi-User TDD Systems, in: Proc. of IEEE International Conference on Communications (ICC), Cape Town, South Africa, 2010, pp. 1–5.
[42] F. Fernandes, A. Ashikhmin, T. L. Marzetta, Inter-Cell Interference in Noncooperative TDD Large Scale Antenna Systems, IEEE Journal on Selected Areas in Communications 31 (2) (2013) 192–201. doi:10.1109/JSAC.2013.130208.
[43] H. Yin, D. Gesbert, M. C. Filippou, Y. Liu, Decontaminating pilots in massive mimo systems, in: Proc. of IEEE International Conference on Communications (ICC), Budapest, Hungary, 2013.
[44] O. Elijah, C. Y. Leow, T. A. Rahman, S. Nunoo, S. Z. Iliya, A comprehensive survey of pilot contamination in massive mimo—5g system, IEEE Communications Surveys & Tutorials 18 (2) (2016) 905–923. doi:10.1109/COMST.2015.2504379.
[45] A. Khan, M. Irfan, Y. Ullah, S. Ahmad, S. Ullah, B. Hayat, Pilot Contamination Mitigation for High and Low Interference Users in Multi-Cell Massive MIMO Systems, in: Proc. of IEEE International Conference on Emerging Technologies (ICET), Peshawar, Pakistan, 2019.
[46] T. Parfait, Y. Kuang, K. Jerry, Performance analysis and comparison of ZF and MRT based downlink massive MIMO systems, in: Proc. of IEEE International Conference on Ubiquitous and Future Networks (ICUFN), Shanghai, China, 2014.
[47] M. Lukoševičius, A Practical Guide to Applying Echo State Networks, Lecture Notes.
[48] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, USA, 1998.
[49] T. Basar, G. J. Olsder, Dynamic Noncooperative Game Theory (Classics in Applied Mathematics), Society for Industrial and Applied Mathematics, USA, 1999.
[50] https://pypi.org/project/simpy/.
[51] http://www.pyopt.org/reference/optimizers.slsqp.html.
[52] https://keras.io/api/optimizers/adam/.
[53] https://keras.io/api/losses/regression_losses/#mean_squared_ error function.
[54] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing. MinMaxScaler.html.
[55] J. Hu, M. McManus, S. K. Moorthy, Y. Cui, Z. Guan, N. Mastronarde, E. S. Bentley, M. Medley, NeXT: A Software-Defined Testbed with Integrated Optimization, Simulation and Experimentation, in: Proc. of IEEE Future Networks World Forum (FNWF): WS5: Federated Testbed as a Service for Future Networks: Challenges the State of the Art, Montreal, Canada, 2022.

Sabarish Krishna Moorthy is a Ph.D. student in the Department of Electrical Engineering at State University of New York at Buffalo (SUNY B̃uffalo), NY, USA. He is currently working in the UB WINGS lab under the guidance of Professor Zhangyu Guan. He received his B.E. degree in Electronics and Communication Engineering and M.S. in Electrical Engineering from Velammal Engineering College, Chennai, India and SUNY Buffalo, NY, USA in 2016 and 2018, respectively. His current research interests are in new spectrum technologies and network design



Nicholas Mastronarde received the B.S. and M.S. degrees in electrical engineering from the University of California, Davis, CA, USA, in 2005 and 2006, respectively, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles, CA, USA, in 2011. He is currently an Associate Professor with the Department of Electrical Engineering, University at Buffalo, Buffalo, NY, USA. His research interests include reinforcement learning, Markov decision processes, resource allocation and scheduling in wireless networks and systems, UAV networks, and 4G/5G networks.

Scott Pudlewski received the B.S. degree in electrical engineering from the Rochester Institute of Technology, Rochester, NY, USA, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from University at Buffalo, The State University of New York (SUNY), Buffalo, NY, USA, in 2010 and 2012, respectively. He is currently a Senior Research Engineer with the Georgia Tech Research Institute (GTRI), Atlanta, GA, USA. His main research interests include network vulnerabilities, the Internet of Things, networking in contested tactical networks, convex optimization, and wireless networks in general.

Elizabeth Serena Bentley received the B.S. degree in electrical engineering from Cornell University, the M.S. degree in electrical engineering from Lehigh University, and the Ph.D. degree in electrical engineering from University at Buffalo. She was a National Research Council Postdoctoral Research Associate with Air Force Research Laboratory (AFRL), Rome, NY, USA. She is currently employed by the AFRL Information Directorate, performing in-house research and development in the Communication Systems and Technology Branch and managing the Aerial Layer Communication Technology (ALCT) program. Her research interests are in cross-layer optimization, swarm networking, directional networking, wireless multiple-access communications, and modeling and simulation.

Zhangyu Guan (SM'11) received the Ph.D. degree in Communication and Information Systems from Shandong University in China in 2010. He is c̆urrently an Assistant Professor with the Department of Electrical Engineering at the State University of New York at Buffalo, where he directs the W̃ireless Intelligent Networking and Security (WINGS) Lab, with research interests in network design automation, new spectrum technologies, and wireless network security. He has served as an Area Editor for Elsevier Journal of Computer Networks since July 2019. He has served as TPC Chair for IEEE INFOCOM Workshop on Wireless Communications and Networking in Extreme Environments (WCNEE) 2020, Student Travel Grants Chair for IEEE Sensor, Mesh and Ad Hoc Communications and Networks (SECON) 2019-2020, Information System (EDAS) Chair for IEEE Consumer Communications Networking Conference (CCNC) 2021. He has also served as TPC member for IEEE INFOCOM 2016-2021, IEEE GLOBECOM 2015-2020, IEEE MASS 2017-2019, IEEE IPCCC 2015-2020, among others.