Generalized Path Planning for Collaborative UAVs using Reinforcement and Imitation Learning

Jack Farley*
jack.farley@duke.edu
Department of Computer Science
Duke University
Durham, North Carolina, USA

Amirahmad Chapnevis
chapnevisa@vcu.edu
Department of Computer Science
Virginia Commonwealth University
Richmond, Virginia, USA

Eyuphan Bulut
ebulut@vcu.edu
Department of Computer Science
Virginia Commonwealth University
Richmond, Virginia, USA

ABSTRACT

Cellular-connected Unmanned Aerial Vehicles (UAVs) need consistent cellular network connectivity to effectively accomplish their designated missions. However, when navigating through regions with partial coverage, such as rural areas, the task of planning the flight paths for these UAV missions becomes notably intricate. Algorithms designed to solve this issue require significant computational resources, making them infeasible for active deployment where an algorithm must run in real time using small compute power. Furthermore, these algorithms exponentially scale in runtime with respect to the number of UAVs being considered. To tackle this problem, we model the parameter space as a discrete grid-world, enable collaboration between drones, and gather supervised data from nonlinear programming and unsupervised data from a simulated version of the environment with associated rewards. We then train a Deep Neural Network (DNN) on this data and approximate optimal results by combining imitation and reinforcement learning methods. This DNN can successfully be deployed at fast speeds using relatively small computational power and can generalize to unseen maps where drone collaboration can be used to reduce mission time. By using the results of a network trained on supervised data as a guiding hand during training, our reinforcement learning approach achieves results better than either method in isolation.

CCS CONCEPTS

• **Networks** → **Mobile networks**; • **Computing methodologies** → **Multi-agent planning**; *Neural networks.*

KEYWORDS

UAV, trajectory optimization, cellular network, reinforcement learning.

ACM Reference Format:
Jack Farley, Amirahmad Chapnevis, and Eyuphan Bulut. 2023. Generalized Path Planning for Collaborative UAVs using Reinforcement and Imitation Learning. In *The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3565287.3617622

*The student was a REU participant at Virginia Commonwealth University when this work was performed.

MobiHoc '23, October 23–26, 2023, Washington, DC, USA

ACM ISBN 978-1-4503-9926-5/23/10…$15.00
https://doi.org/10.1145/3565287.3617622

1 INTRODUCTION

Unmanned Aerial Vehicles are used in a variety of applications, such as 3D mapping [12], environmental monitoring [6], natural disaster monitoring and prediction [1], and traffic monitoring [14]. For UAVs to be used practically, they must have secure and consistent connection. Most off-the-shelf UAVs rely on line of sight communication, significantly decreasing the range of UAV flights, thus restricting their potential usage. As a result, recent attempts have been made to utilize cell towers for connection, to ensure that UAVs can maintain connections over longer and more complex flights. However, finding the path of UAVs, particularly in rural areas with a sparse connectivity, is a significant challenge.

Recent efforts have incorporated an acceptable duration for continuous network outages in the context of UAVs [3] and focused on optimizing the UAV trajectories considering this outage constraint. The objective is to reduce the overall mission completion time for all UAVs, ensuring that none of them encounter a connectivity outage surpassing a predefined threshold. A collaboration between UAVs has also been considered in these studies, enabling the UAVs to serve as relays to each other to maintain connections and curtail their flight paths.

In order to solve this problem, a variety of tools [7] have been utilized in existing studies [3, 16]. However, many of these solutions scale quite poorly in run time with respect to the number of UAVs being utilized, or they often require extensive information about the state when planning paths. This study seeks to outline a potential method for fast and generalized collaborative path planning under this outage constraint.

The rest of the paper is organized as follows. We discuss the related work in Section 2 followed by a system model described in Section 3. In Section 4, we then outline how we have created the data used by our model. In Section 5, we outline the specifics of training and network initialization. In Section 6, we provide numerical results and compare approximations to optimal results. Finally, we conclude and discuss future work in Section 7.

2 RELATED WORK

There have been several recent contributions to the field of trajectory planning for UAVs. In [3], the authors explore the trajectory optimization for a cellular-connected UAV considering the outage constraint for the first time, followed by the studies in [5, 7, 16]. In

Notations	Description
$\mathcal{U}, \mathcal{G}$	The set of UAVs and GBSs
n, m	Number of UAVs and GBSs
L_S^u, L_F^u	Start and final location of UAV u
$x_u(t), y_u(t)$	Location of UAV u in timeslot t.
R_g	The range of GBS ⟶ UAV connection
R_u	The range of UAV ⟶ UAV connection
T_{Total}	Sum of flight durations of all UAVs
o_{max}	Maximum continuous outage threshold for UAVs
$\mathcal{M}$	Nodes (points) created on the map
$\mathcal{M}(a)$	a-th point on the map
$Q_M(A, t)$	Minimum distance of UAV M from the start point when it is located in node A at time slot t
$\mathcal{N}(a)$	Set of nodes that has at least one direct path to node a (Neighbours of Node a)
$out_u(a, t)$	Outage time for UAV u when it is located in node a at time slot t

Table 1: Notations and their descriptions.

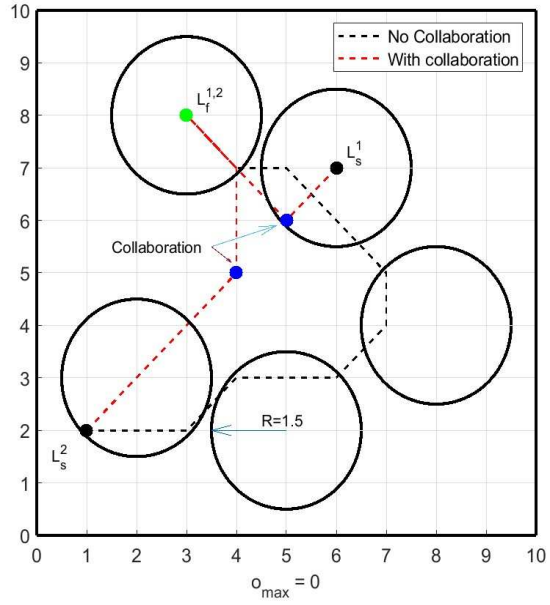

Figure 1: An example of optimal drone paths with and without collaboration.

[4], collaboration among UAVs is also considered as part of the path planning algorithm. In [12], a generalized algorithm with quick run time is developed relying on having significant information about the surrounding terrain.

The field of multi-agent path planning (MAPF) has also exploded in recent years with the availability and success of deep learning. In [13], reinforcement learning (RL) and imitation learning (IL) are combined to optimize collision avoidance in path planning and to improve the run time performance. In [8], a new and improved multi-agent actor-critic solution is proposed using centralized value approximation and decentralized policies and a performance improvement in several benchmark domains has been shown. Despite the variety of studies in trajectory planning for UAVs, to the best of our knowledge, there is no study that specifically focuses on generalized path planning for collaborative UAVs in the context we defined in this work. Similarly, within the MAPF literature, there is a limited set of works that overlap partially with the problem studied here. For example, in [10], a generalized solution is targeted including many agents, however the study does not consider active collaboration among UAVs as we consider in this study.

3 SYSTEM MODEL

For a given number of UAVs, n, we denote the set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ as the set of UAVs. The location of each UAV in $\mathcal{U}$ is defined by an x and y coordinate at each time step, t, by $u(t) = (x_u(t), y_u(t))$. We denote the set of m different ground base stations (GBSs) or cell towers by $|\mathcal{G}| = m$, where $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$. The location of each GBS in $\mathcal{G}$ is defined by an x and y coordinate, $g_i = (x_i, y_i)$. We assume a discrete 2 dimensional *grid world* for our experiments, where all coordinates in the system are integers. Each UAV can take 1 of 9 actions at each time step: a single movement to any adjacent space including diagonals or no movement. All GBSs are assumed to have the same range of connectivity, R_g. Any UAV within distance R_g of any GBS at time t is considered connected at time t. All UAVs are able to act as signal extenders to any other UAV within range R_u. This signal extension can be chained together. The goal of the system is to find a path that takes each UAV from its respective start point L_s to its respective end point L_f while primarily minimizing total mission time (Fig. 1). Along this path, no UAV can be continuously disconnected for more than a maximum outage time denoted by o_{max}. Mission time is defined as the maximum of the set T, where the set T is a list of all individual times taken for each drone to traverse from its start point to destination. This system model is similar to the one in [4].

Table 1 shows the notations used throughout the paper. Fig. 1 shows an example of UAV flight utilizing collaborative planning.

4 DATASET GENERATION

4.1 Random Map Creation

We generated a set of 14,000 random maps using a 10 by 10 grid. All maps are restricted to 2 UAVs. Maps have either 4 or 5 GBSs and have an outage constraint of either 0 or 1. These values are selected in an attempt to sample maps that would include a higher sample of collaboration, as maps with high outage and/or many GBSs would often permit paths directly from start to destination for both UAVs. We randomly created maps by placing m number of GBSs randomly on the map. If any GBSs has more than a certain threshold of intersection with another (i.e., %20), it was replaced. Then, start and end points were selected randomly until they were both within range of a GBS. Given these start and end points, a feasibility check was performed to ensure all drones could reach the destination within some time horizon. If the map was infeasible, the process was restarted. If the map was deemed feasible, the

Algorithm 1 Finding Individual UAV path (u) and check the feasibility

Input: L_S^u,L_F^u, $\mathcal{M}$,$\mathcal{T}$
$Q_u(a,t) \leftarrow \infty \; \forall t \leq \mathcal{T},\ a \in \mathcal{M}$
$Q_u(L_S^u, 0) \leftarrow 0$
for t in $1 : \mathcal{T}$ **do**
  for $a = 0$ *to*$|\mathcal{M}|$ **do**
    for $\forall\ a' \in \mathcal{N}(a)$ **do**
      if $Range_G(a')$ **then**
        if $Q_u(a, t-1) + dist(a, a') < Q_u(a', t)$ **then**
          $Q_u(a', t) \leftarrow Q_u(a, t-1) + dist(a, a')$
          $out_u(a', t) \leftarrow 0$
          $\mathcal{M}(a', t) \leftarrow a$
        end if
      else
        if $out_u(a, t-1) < o_{max}$ **then**
          if $Q_u(a, t-1) + dist(a, a') < Q_u(a', t)$ **then**
            $Q_u(a', t) \leftarrow Q_u(a, t-1) + dist(a, a')$
            $out_u(a', t) \leftarrow out_u(a, t-1) + 1$
            $\mathcal{M}(a', t) \leftarrow a$
          end if
        end if
      end if
    end for
  end for
end for
if $Q_u(L_F^u, \mathcal{T}) \neq 0$ **then**
  return *Feasible Path*
end if
return *Unfeasible Path*

map was saved and added to the dataset. The dynamic programming (DP) algorithm for feasibility is given in Algorithm 1 (input $\mathcal{T}$ is a time constraint, and $Q_u(a,t)$ denotes the distance of UAV u from its destination when it is located at point a, along a path that does not break the o_{max} at time t):

4.2 Nonlinear Programming

In previous work [4], this problem is successfully solved and modeled using nonlinear programming. While quite slow, these results are guaranteed to find optimal paths under constraints. To create a supervised dataset, we run the nonlinear optimization on all randomly created maps to create a dataset of optimal behavior for our model to learn from. Each map takes roughly 90 seconds to converge to an optimal solution. The definition of our nonlinear optimization can be found in [4].

4.3 Reinforcement Learning

We use the classical formulation of a Markov Decision Process (MDP) [2] to collect episodes of experience. We define an agent, environment, a set of states {S}, a set of actions {A}, and policy π. We collect reward at each state-action transition. We assume this process to have the Markov property.
We define a state s to include:

1. The current coordinates of UAV u at time t, $(x_u(t), y_u(t))$
2. The coordinates of all other UAVs, $(x_i(t), y_i(t)), \forall i \neq u$ where $i \in \mathcal{U} \neq u$
3. The coordinates of all GBSs $(x_j, y_j) \forall j \in \mathcal{G}$
4. The outage constraint o_{max}
5. The matrix of boolean vector representing current connectivity for all UAVs $\mathbb{C}$
6. The integer vector representing current continuous outage for all UAVS C_{out}, where $\mathbb{C}_{out}[i] = 0$ if $\mathbb{C}[i] = 1$
7. The Range of UAV → UAV connection R_U
8. The range of GBS → UAV connection R_G

We take an action A at one of 9 choices, either jumping to a point on the grid that is adjacent including diagonal points, or standing still.

We collect reward according to specifics outlined later in this paragraph.
A state S_t is Markov iff:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, ..., S_t]. \tag{1}$$

We assume the Markov property $\forall S_t \in \{S\}$. The MDP defines:

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]. \tag{2}$$

The policy π is defined:

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s] \forall a \in A, \forall s \in S, \tag{3}$$

with expected reward:

$$\mathcal{R} = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]. \tag{4}$$

We then seek to find a policy π that optimizes this expected reward $\forall S_t \in \{S\}$. We can define a state value function $v_\pi(s)$ starting from state s following policy π:

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s], \tag{5}$$

where G_t is expected cumulative reward. We define a state-action function as the expected reward starting from state s, taking action a, and then following policy π:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]. \tag{6}$$

We consider future states to have discounted reward, γ, i.e., a reward x steps in the future should be weighed $\gamma^{(x-1)}$ as heavily as an immediate reward:

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]. \tag{7}$$

We use a discount factor of 0.99, as is common practice. This state-value equation is then solved recursively.

Historically, these values are tracked with a table of all state-action pairs [2], manually updating the average reward received in the steps following the execution of action a in state s. However, a DNN can be used to perform complex function approximation, to update these values based on learned experiences. This is a standard reinforcement learning algorithm referred to as deep Q learning (DQN) [11].

We collected experiences by simulating this *grid world* environment, having agents initialized with $(x_u(0), y_u(0)) = L_s^u, \forall u \in \mathcal{U}$ for each random map. Agents act according to policy π, which iteratively improves. Agents receive a reward as in [13] following:

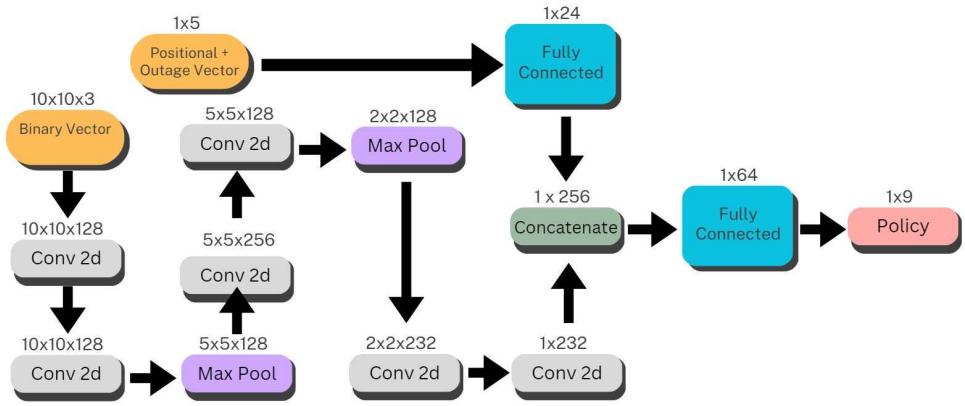

Figure 2: A flowchart documenting the DNN architecture.

$$R_{t+1} = \delta d - p - 1$$
$$dist(a, b) = \sqrt{(a[0] - b[0])^2 + (a[1] - b[1])^2)}$$
$$\delta d = dist(S_{t-1}, L_f) - dist(S_t, L_f)$$

where p = -0.25 if action is no movement and $S_t \neq L_f$.

If all agents reach their respective destinations, all agents receive a reward of +10, and the episode ends. If an episode lasts a certain amount of time t_{stop} without a destination being found, the episode is terminated, and all agents receive no further reward. Experiences are collected in parallel, as the environment supports an arbitrary number of agents. In training, agents are unable to access states that violate the outage constraint. At each step, each agent is checked if it is in range of a GBS. If an agent is in range of a GBS, the agent's current location is stored as a variable. When the agents violate the outage constraint, they are transported back to the last index where they were in within range of a GBS. This greatly increases the training efficiency, as compared to negative rewards for violating the outage constraint. This approach has the drawback of not teaching agents how to properly recover when they violate the outage constraint, but, in practice, most algorithms would require agents to simply fly back to the last safe location after violating the outage constraint. Agents do not share observations but act according to the same policy network. The policy network then learns from the combined experience of all agents. We ran approximately 45,000 episodes of experience collection, accounting for each map being sampled 4 times of the 80% of the random maps that are in the training set. Maps are sampled randomly.

For one RL network, we randomly initialize the policy network. As per [11], we exponentially decay the probability of random actions, ϵ, from 0.9 to 0.05 over the first 60,000 time steps of training. This selection of random actions naturally allows the agent to explore sufficiently and to gain meaningful information as it refines its policy. This decay is standard practice for DQN implementation.

For the other network, we use the network trained on supervised data as a *guide*, similar to [13]. We exponentially decay the probability of random actions, ϵ, from 0.2 to 0.05 over the first 60,000 time steps of training to ensure some exploration is still done. However, if an action is not selected at random, we select the action using the *guide* network with some probability. This probability starts at 0.6 and linearly decays to 0 over the first 60,000 time steps.

5 NET AND TRAINING SPECIFICS

5.1 Observation and DNN Structure

We transformed observations into normalized coordinates and binary matrices, which is similar to the structures used in [13]. We converted GBS coordinates into a binary 10 by 10 matrix of connectivity. We converted the positions of other UAVs into two 10 by 10 binary matrices- one matrix representing the current positions of other UAVs, and the other representing goal positions of the other UAVs. These three matrices are concatenated into a 10 by 10 by 3 matrix, representing the state information independent of the drone's position. We also create a normalized matrix, representing the current position of the UAV in question, the goal of the current UAV, and the outage constraint o_{max}. The 10 by 10 by 3 matrix is passed through 2 convolutional layers, followed by a maxpool, followed by two subsequent convolutional layers with a final maxpool, and flattened. The 5 by 1 normalized vector is fed into a fully connected layer (FCL), the output is rectified, and the output of this FCL (128 by 1) is concatenated with the other flattened output. This concatenated vector is fed into two subsequent fully connected layers that output to the final policy (9 by 1), which represent the nine possible actions at any time step. This is a very similar network structure used in [13], which was inspired by [15]. This way of creating observations naturally generalizes to any arbitrary number of UAVs. We show a visualization of this model in Fig. 2.

5.2 Loss and Parameters

For both models, we used the AdamW optimizer. For imitation learning, we used the cross-entropy loss between output action probabilities and ground truth actions taken by agents in the ILP dataset. For reinforcement learning, we used the Huber loss [9] between expected and actual state-action values. We experimented

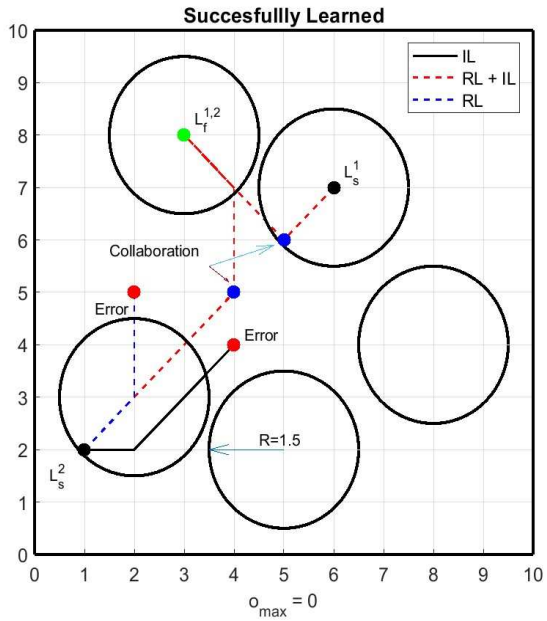

Figure 3: RL + IL successfully learns collaboration.

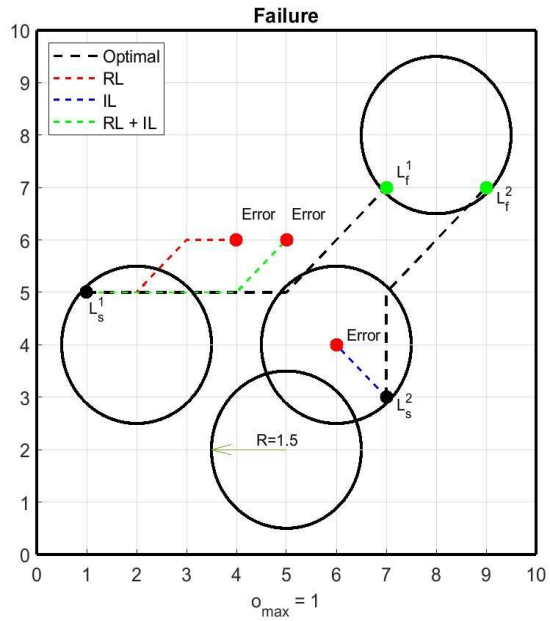

Figure 4: No policy generalizes correctly.

Table 2: Distribution of accurate, sub-optimal and failure paths considered during training of different solutions.

Results	IL	RL	RL with IL
Accurate	65.0%	56.7%	70.2%
Sub optimal	15.5%	12.8%	8.5%
Critical Failure	19.5%	30.5%	19.4%

Table 3: Percentage of correct results that included collaborative behavior for each learned policy.

Results	Baseline	IL	RL	RL with IL
Collaboration	16.8%	8.3%	3.4%	11.4%

with different hyperparameters but generally used a learning rate near 1e-4.

5.3 Training

For imitation learning, we trained the model on 80% of the generated data, holding the other 20% for testing. We found generalization to be optimized using around 8-15 epochs. Before each epoch, we randomly shuffled samples as is standard practice.

For reinforcement learning, we trained the model on the same 80% of maps for consistency when comparing and transferring results. The resulting data points are inherently different when compared to the supervised case, as the agent can only sample experiences using its policy (it of course learns off policy). For both models we used a batch size of 128. We updated the target network every fifth batch.

6 RESULTS

We tested our learned policies on ~2,000 maps that were not included in training. We used the same structure that we did when gathering RL data. That is, we initialized UAVs at their respective start points and had agents act according to the policy they had learned in training. We defined accurate trials as trials where both UAVs successfully navigated to the destination, without violating o_{max}, with mission time equivalent to that of the optimal baseline run on the same map. We defined sub-optimal trials as trials where both UAVs successfully navigated to the destination without violating o_{max}, with: optimal < mission time < t_{stop}. We defined critical failures as trials where the outage constraint was violated (upon violating o_{max}, trials were terminated immediately) or where t_{stop} time passed without both drones successfully navigating to there destinations. Importantly, we lifted the safety guards used in training that would restrict agents from violating the outage constraint to be able to observe how they would behave in a realistic scenario.

Table 2 shows the distribution of different type of paths used during training of different solutions. As the results show, the network trained using RL + IL is the most accurate of the three. The network seems to have the benefit of being guided down promising paths by its supervised guide and then fully exploring those paths using reinforcement learning. Given a network is stored onto the system, results are quite fast, with the 2000 trajectories being generated in less than 5 seconds. The stored network is on the scale of megabytes of storage. This hints at the feasibility of using this algorithm in live situations with storage and time constraints.

To measure the extent to which collaboration was learned, we also calculated the percentage of correct results that included collaborative behavior for each learned policy. We define collaboration as any trial that includes a time step where one of the two UAVs is connected to a GBS, but one of the UAVs is not in range of a GBS, but rather in range of the other UAV. We define baseline as the percentage of optimal runs in the test set that included collaboration. As it is shown in Table 3, our RL + IL policy did the best job in successfully learning how to collaborate. An example of this is shown in Fig. 3 . It appears from both of these results that the RL + IL policy successfully learned a deeper and more complex policy when compared to its two counterparts. It also suggests that this policy was better at finding optimal routes in complex maps, as maps that need collaboration are inherently more likely to require complex behaviors.

For all policies, collaboration was undersampled in correct runs when compared to baseline. This is intuitive, as collaborative policies are relatively harder to learn because they require coordinated behavior.

Looking at unsuccessful runs, it seems evident that the policy learned through IL acts more conservatively. In its critical failures, it more frequently fails by sitting still rather than violating the outage constraint. An example of this is shown in Fig. 4. Both RL algorithms violate the outage constraint more frequently, although RL + IL is an upgrade over RL alone in almost every way. It seems RL + IL vs. IL is a bit more of a complex comparison, as sitting still could be considered a less harmful failure than going astray. Additionally, although the RL + IL algorithm behaves optimally more often, both algorithms have roughly the same failure rate, due to the higher rate of sub optimal routes completed by the IL policy. Asymptotically with respect to grid size and number of agents, we believe that the combination of IL + RL would prove much more fruitful than plain IL, given that it is difficult to generate a meaningfully sized dataset of only IL outcomes. This asymptotic improvement has been demonstrated by [8], [13], and others.

It also seems evident that our model learned from too many trivial examples. It is a goal of future work to generate a dataset that more heavily contains complex and collaboration focused behaviors.

From a technical perspective, it is meaningful to mention the number of parameters (observation structure, network, algorithm choice, algorithm specific parameters, training specifics, etc.) that could have potentially increased or decreased performance. Many were tested, but given the length of time needed to train models (RL models specifically), we only were able to explore a small fraction of this potential parameter space.

7 CONCLUSION

Cellular-connected Unmanned Aerial Vehicles require continuous cellular network connectivity to effectively complete their missions. However, in areas with limited coverage, like rural regions, planning flight paths for these UAV missions becomes complex. Recent endeavors have aimed to optimize UAV trajectories within acceptable network outage periods, enhancing mission completion times. The goal is to prevent UAVs from experiencing connectivity outages beyond a predefined limit. Collaborative UAV efforts involve using some UAVs as relays to maintain connections and shorten flight paths. Nonetheless, the computational demands of solving this challenge hinder real-time deployment, especially with multiple UAVs. To address this, we create a model using supervised and unsupervised data, training a Deep Neural Network (DNN) through a combination of imitation and reinforcement learning. This DNN can perform effectively with limited computation power and adapt to new environments. Through simulations, we have shown that our approach that merge supervised and reinforcement learning outperforms both individual methods.

ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation (NSF) Award# 2050958: REU Site: End-User Programming of Cyber-Physical Systems and Award# 1815603. 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 the funding agencies.

REFERENCES

[1] Stuart M Adams and Carol J Friedland. 2011. A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management. In *9th international workshop on remote sensing for disaster response*, Vol. 8. 1–8.
[2] Barto and Sutton. 1992. *reinforcement Learning: An Introduction*. http://incompleteideas.net/book/RLbook2020.pdf
[3] Eyuphan Bulut and Ismail Guevenc. 2018. Trajectory optimization for cellular-connected UAVs with disconnectivity constraint. In *IEEE International Conference on Communications Workshops (ICC Workshops)*. 1–6.
[4] Amirahmad Chapnevis, Ismail Güvenç, Laurent Njilla, and Eyuphan Bulut. 2021. Collaborative trajectory optimization for outage-aware cellular-enabled UAVs. In *IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. 1–6.
[5] Yu-Jia Chen and Da-Yu Huang. 2020. Trajectory optimization for cellular-enabled UAV with connectivity outage constraint. *IEEE Access* 8 (2020), 29205–29218.
[6] AS Danilov, Ur D Smirnov, and MA Pashkevich. 2015. The system of the ecological monitoring of environment which is based on the usage of UAV. *Russian journal of ecology* 46, 1 (2015), 14–19.
[7] Omid Esrafilian, Rajeev Gangula, and David Gesbert. 2020. 3D-map assisted UAV trajectory design under cellular connectivity constraints. In *IEEE International Conference on Communications (ICC)*. 1–6.
[8] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
[9] Kaan Gokcesu and Hakan Gokcesu. 2021. Generalized huber loss for robust learning and its efficient minimization for a robust statistics. *arXiv preprint arXiv:2108.12627* (2021).
[10] Nir Greshler, Ofir Gordon, Oren Salzman, and Nahum Shimkin. 2021. Cooperative multi-agent path finding: Beyond path planning and collision avoidance. In *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 20–28.
[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[12] Francesco Nex and Fabio Remondino. 2014. UAV for 3D mapping applications: a review. *Applied geomatics* 6 (2014), 1–15.
[13] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. 2019. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters* 4, 3 (2019), 2378–2385.
[14] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. 2019. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *Ieee Access* 7 (2019), 48572–48634.
[15] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[16] Shuowen Zhang and Rui Zhang. 2019. Trajectory design for cellular-connected UAV under outage duration constraint. In *IEEE International Conference on Communications (ICC)*. 1–6.