Deep Reinforcement Learning for UAV-Assisted Emergency Response

Isabella Lee*
University of Illinois at Urbana-Champaign
ile@illinois.edu

Vignesh Babu
University of Illinois at Urbana-Champaign
babu3@illinois.edu

Matthew Caesar
University of Illinois at Urbana-Champaign
caesar@illinois.edu

David Nicol
University of Illinois at Urbana-Champaign
dmnicol@illinois.edu

ABSTRACT

In the aftermath of a disaster, the ability to reliably communicate and coordinate emergency response could make a meaningful difference in the number of lives saved or lost. However, post-disaster areas tend to have limited functioning communication network infrastructure while emergency response teams are carrying increasingly more devices, such as sensors and video transmitting equipment, which can be low-powered with limited transmission ranges. In such scenarios, unmanned aerial vehicles (UAVs) can be used as relays to connect these devices with each other. Since first responders are likely to be constantly mobile, the problem of where these UAVs are placed and how they move in response to the changing environment could have a large effect on the number of connections this UAV relay network is able to maintain. In this work, we propose DroneDR, a reinforcement learning framework for UAV positioning that uses information about connectivity requirements and user node positions to decide how to move each UAV in the network while maintaining connectivity between UAVs. The proposed approach is shown to outperform other greedy heuristics across a broad range of scenarios and demonstrates the potential in using reinforcement learning techniques to aid communication during disaster relief operations.

CCS CONCEPTS

• **Networks** → **Wireless access points, base stations and infrastructure**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

UAV network, reinforcement learning, IoT network, disaster relief

ACM Reference Format:
Isabella Lee, Vignesh Babu, Matthew Caesar, and David Nicol. 2020. Deep Reinforcement Learning for UAV-Assisted Emergency Response. In *17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous), December 07–09, 2020, N/A, Cyberspace.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

*Now at Google

MobiQuitous '20, December 07–09, 2020, N/A, Cyberspace

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

When a natural disaster strikes, it not only has the potential to destroy homes, but can also damage critical communication infrastructure, such as the case with the 2019 Cyclone Idai in Southern Africa [1] and Typhoon Phanfone in the Philippines [2]. For first responders, a solid communication channel between emergency response groups could have a drastic impact on the efficiency and efficacy of their work, potentially resulting in many more lives being saved. Specifically, it is important for lead officers and commanders to have reliable communication channels with other leaders across emergency response groups as well as with all the responders within their group.

Unmanned aerial vehicles (UAVs), also known as drones, can be key enablers of communication in such situations. One of the greatest benefits of a UAV-assisted network is that it can be quickly deployed and adapted according to a changing post-disaster environment as opposed to other methods of disaster relief communications, such as installing Wi-Fi APs throughout the affected area [3], which requires significant time and manual labor. In the past, UAVs have been used to construct communication networks on-demand during first phases of disaster recovery [4]. UAVs have also aided in forming an emergency Wi-Fi network [5] as well as been mounted with LTE femtocell base stations and used to provide cell coverage in areas struck by large-scale disasters [6].

However, there are many challenges involved in operating a fleet of UAVs in such uncertain environmental conditions. Depending on channel conditions and availability of satellite backhaul links, drones may need to remain within a certain range of each other in order to communicate. Amidst the wrecked buildings and fallen infrastructure, there may not be direct line of sight to first responders, who could change positions frequently as they move around to carry out disaster relief operations. Since a drone's location determines its ability to provide connectivity to a particular user, the trajectory of each drone in the UAV network can have a significant impact on the utility of the overall communication network.

In this paper, we are interested in answering the question: **How should a UAV relay network re-position itself in response to mobility of first responders (users) on the ground while maintaining operation-specific connectivity requirements?**

To address this problem, we take a reinforcement learning (RL)-based approach. In disaster relief scenarios, where first responders move according to mission-specific requirements and follow complex movement patterns [7], a RL-based approach can prove advantageous compared to a traditional machine learning approach because it can implicitly learn any existing mobility patterns and make decisions by observing the current state of the system while simultaneously accounting for future state transitions.

We develop a framework called DroneDR (short for Drone-aided Disaster Relief), which uses reinforcement learning to intelligently position UAVs serving as airborne relays for groups of first responders. As is typical [8], we assume that each group of first responders is assigned a leader to coordinate tasks and monitor other group members. DroneDR continuously observes current locations of all groups of first responders at discrete timesteps and strives to position drones to maintain connectivity among them. In particular, DroneDR strives to ensure that: (1) data generated by group members is gathered at each group's leader (2) communication between different group leaders is enabled opportunistically.

Drones in our proposed framework are constrained to move as a strongly connected group while attempting to maximize the number of established connections. A connection is established between two first responders if and only if both of them have a functioning uplink to at least one drone. We group all IoT devices carried by a particular first responder into a single communication entity that represents that first responder. For the rest of this paper, we simply refer to these entities as *nodes*.

An operating environment could be defined by several parameters such as the total affected area, number of available drones, number of employed nodes, and radio ranges of transmitters installed on both classes of mobile entities: nodes and drones. To demonstrate our algorithm's ability to adapt to different types of operating conditions, we created simulations of multiple environments where all of the above stated parameters were varied. In particular, we considered two classes of disaster-struck environments: (1) a small-scale 100x100m area and (2) a large-scale 1sq km urban area. Through simulations, we show that DroneDR is able to cover more nodes and maintain more connections on average compared to other baselines algorithms which do not employ predictive machine learning techniques. For all operating conditions, DroneDR was able to maintain up to 2x more connections on average than other baselines. DroneDR was also able to maintain more functioning uplinks to nodes (coverage) on average, at times providing 30% more coverage compared to other baseline algorithms.

The rest of this paper is organized as follows: Section 2 describes the overall architecture of DroneDR, Section 3 describes our proposed reinforcement learning algorithm, Section 4 presents the environmental setup, and Sections 5, 6, 7, and 8 form the Evaluation, Discussion, Related Work, and Conclusion of our paper, respectively.

2 BACKGROUND AND SYSTEM ARCHITECTURE

Since first responders and UAVs are mobile entities, the underlying network formed by them evolves over time. The time varying network may be observed at discrete timesteps and classic optimization techniques could be applied to make locally optimal decisions. However, these actions may be suboptimal (as we later show) since they do not plan for future state transitions. Instead, we adopt the use of Deep Reinforcement Learning (DRL) techniques to implicitly learn how the network evolves over time and recommend the best actions to take at each timestep based on this learned knowledge. In reinforcement learning, at each timestep, a RL agent observes the current state of the system and uses a learned policy to decide on an action. This drives the system to a new state and the agent receives a reward signal according to how the action affected the state.

In this work, we model each disaster-struck environment as a Markov Decision Process (MDP), which includes a state space S, an action space A, a reward signal $r(S, A)$, and environment dynamics $r(S_t, A_t) : S \times A \rightarrow R$ describing the execution of action A_t when the state is S_t and the resulting feedback from the environment. Using DRL, the aim is to learn an optimal policy $A_t = \pi(S_t)$ (mapping states to actions) for drone placement where the environment dynamics are hidden from the agent.

The overall architecture of DroneDR is illustrated in Figure 1. It comprises a set of drones, nodes and a central controller implementing the intelligent positioning algorithm. Each node and drone is assumed to be fitted with a GPS tracker and their positions are gathered at the central controller in real time. Computations performed based on the gathered coordinates are forwarded to one of the drones, which is assumed to have a stable long distance backhaul link, for eventual distribution across the strongly connected drone network.

Learning how to best position drones among a theoretically infinite set of possible locations is not only challenging, but it could also lead to non-optimal behavior in the real world if the algorithm is sensitive to fine-grained changes in UAV and ground node positions. This is because the current state of the system is measured at discrete intervals and mission-specific noise could be introduced into gathered measurements based on events that occurred in between two state measurements. Some of these events (e.g GPS inaccuracies or drones being displaced from their desired positions by wind) may be extraneous to the modelled environment and beyond the control of the positioning algorithm. It is difficult to incorporate these sources of noise into the simulated training environment.

Thus, to desensitize the algorithm to fine-grained changes in drone and node positions, we discretize ground and airspace into fixed-size grids and map collected coordinates to their holding grid cells. This operation is performed by the Discretizer module depicted in Figure 1. Hence, our proposed algorithm observes the current *locality* (or grid) of each ground sensor and drone and guides placement of drones to suitable localities. It is impervious to events that may happen within each grid. Furthermore, drone movement is constrained to a fixed height in each environment and therefore, each drone only moves in the x-y-coordinate plane. This limitation on degrees of freedom of drone movement was made in favor of using a realistic environment model (described in Section 4) which involved radio frequency ray tracing at a fixed height. We leave the range of possible drone altitude changes for future

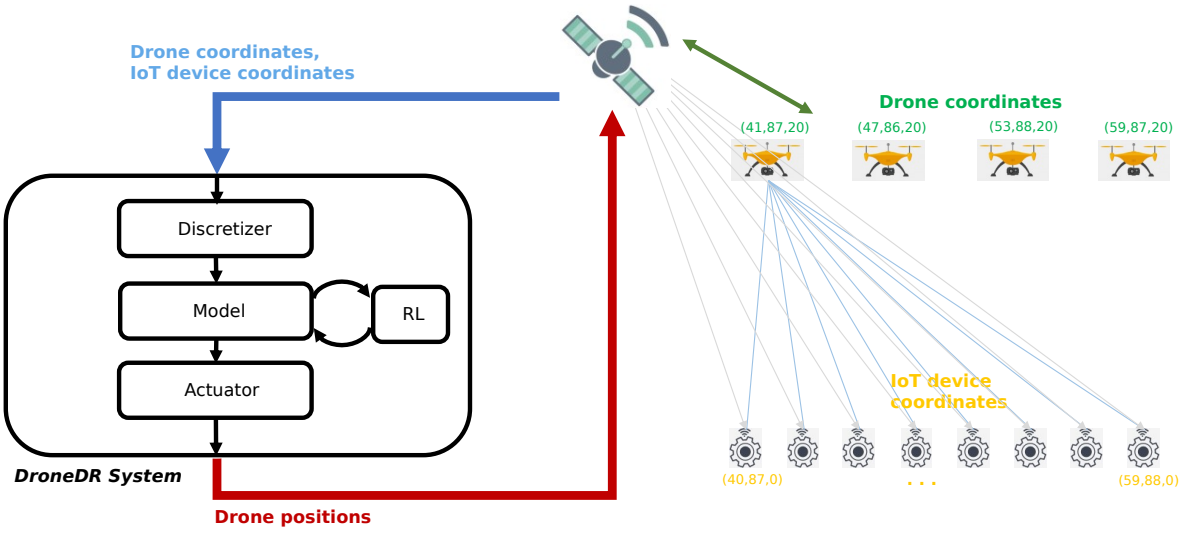

Figure 1: DroneDR: System architecture

work and describe the drone movement algorithm in more detail in Section 3.

3 SYSTEM MODEL

Our system takes as input the number of drones (M), a set of node group descriptions ($G = \{H_i\}$ where H_i represents the size of the i^{th} group), a node radio range, and a drone radio range (which specify how far apart neighboring entities can be in order to still be connected to each other). It outputs a trained policy that recommends the best drone positions at each system state.

From the specified inputs, without loss of generality, DroneDR associates a leader role for each node in $\{H_{i0} \ \forall \ i\}$ and a non-leader role for other nodes. DroneDR strives to establish connectivity within each group (i.e between all leader to non-leader pairs $(H_{i0}, H_{ij}) \ \forall j \neq 0, \forall i$) and across all group leaders (i.e between all pairs $(H_{i0}, H_{j0}) \ \forall \ i \neq j$).

3.1 State Space

The RL agent interacts with its environment by observing the state of the system and its transitions at discrete timesteps. At time step t, we use the following observation vector to describe the current state of the system:

$$S_t = \{(U_{1_t}, T_{1_t}, C_{1_t}) \dots (U_{N_t}, T_{N_t}, C_{N_t}), \\ (D_{1_t}, Q_{1_t}) \dots (D_{M_t}, Q_{M_t})\} \quad (1)$$

where $N = \sum_{k=1}^{|G|} H_k$ is the total number of nodes, T_{j_t} is the amount of time spent by the j^{th} user node at its location, C_{j_t} is a boolean connection status of the j^{th} node (which indicates whether the node is currently connected to at least one drone) and Q_{i_t} indicates the number of nodes that are currently connected to the i^{th} drone. U_{j_t} and D_{i_t} represent the cartesian (x,y,z) coordinates of the centers of the discretized fixed size grids occupied by the j^{th} node and i^{th} drone at time t respectively.

The connection status of each node is determined by the relative position of the node with respect to all drones. Each node is associated with a communication range (r_n meters) which specifies the maximum allowed distance between the grids occupied by a node and a drone for a functioning connection between the pair of entities. Similarly, each drone is also associated with a communication range (r_d meters) which specifies the maximum allowed distance between the grids occupied by two drones for them to be connected. These parameters are specific to each environment and included with an environment description. In Section 4, we discuss how environments are specified in greater detail.

3.2 Action Space

The action space at timestep t is described by the following vector:

$$A_t = \{a_{1_t}, a_{2_t}, \dots, a_{M_t}\} \quad (2)$$

and a_{i_t} represents one of the user nodes. Essentially, the action to be taken at each time step is presented as a list of node IDs. This list is M elements long so that each drone can be assigned to a unique node in the specified list of nodes to move towards. However, a drone will only take the step towards a node if that movement does not disconnect the drone network. In other words, a strongly connected drone network graph (where every drone can reach every other drone through at least one path in the graph) is maintained at all times. Drone-to-node assignments are made based on each drone's location-based proximity to the target nodes as well as the nodes' role (i.e. whether or not they are leader nodes). Pseudo-code for this approach is shown in Algorithm 1. At every iteration, the *takeAction* function, which takes the action space as input, is called.

Ideally, all of the drones will be able to move one step closer to their assigned nodes without moving too far apart from each other (and thereby disconnecting the drone network), but it is also possible that no such movements are feasible so the whole group of drones will move in the same direction in order to maintain inter-drone connectivity. Even if the two drones are out of direct range of each other, the movement algorithm defined in Algorithm 1 ensures

```
function tryMovingDrones(dronePool, targetPos):
    dronePos[L_1...L_M] = originalDronePos[L_1...L_M]
        // initialize new drone positions as their
        current positions

    sortedDronePool[D_1...D_M] = sort(dronePool) // Sort
        drones in dronePool in increasing order
        based on their distance from targetPos

    // Move closest drone towards targetPos
    droneIdx = sortedDronePool[0]
    dronePos[droneIdx] =
     moveDroneTowardsTargetPos(dronePos[droneIdx],
     targetPos)
    δ = dronePos[droneIdx] - originalDronePos[droneIdx]
    i = 1
    while drone network is disconnected do
        // Move drones in same direction as the
            first drone that moved
        droneIdx = sortedDronePool[i]
        dronePos[droneIdx] = dronePos[droneIdx] + δ
        i = i + 1
    end
    if drone network is still disconnected then
        dronePos = originalDronePos // Undo all the
            moves
    end
    return list of unmoved drones

function takeAction(A[n_1...n_M]):
    nodePool[n_1...n_M] = sort(A[n_1...n_M]) // Sort nodes
        in order of leader nodes first
    dronePool[d_1...d_M]
    for n in nodePool do
        nodePos = getNodePosition(n)
        dronePool = tryMovingDrones(dronePool, nodePos)
            // Only move drones that haven't
            already been moved
        if length(dronePool) < 1 // No drones left to
            move
         then
            break
        end
    end
```

Algorithm 1: Moving drones based on given actions. Action to be taken at each time step is presented as a list of node IDs, which are then sorted according to node roles (i.e. leader or non-leader). Drones are moved in order of location proximity to sorted nodes specified in action space and connectivity amongst drones is always maintained. Connectivity is assured by checking that each node in the pool has a path to the leader node via Depth-first Search.

that a strongly connected drone graph is always maintained.

3.3 Reward Function

The reward function defines how the reinforcement learning agent will be rewarded at the current timestep given the current state after taking a specific action. At timestep t, in response to an action, the total reward provided to the agent is described by the following expression:

$$r(S_t, A_t) = P_t/K \tag{3}$$

where P_t is the number of established connection pairs at time t and K is the maximum number of possible connections. A connection is "established" if and only if both endpoints of the connection have a positive connection status (which is defined in Section 4).

3.4 Proximal Policy Optimization

The specific RL training algorithm we used was Proximal Policy Optimization (PPO), which is a policy gradient method that performs each policy update using multiple epochs of stochastic gradient ascent [9]. PPO has many strengths over other policy gradients such as Trust Region Policy Optimization (TRPO) and Actor Critic with Experience Replay (ACER) since it is more general, has better sample complexity, and is much simpler to implement [9]. The main idea of PPO is to use clipping to ensure that new policies do not deviate too much from old policies after an update [10]. This process helps combat some of the challenges of getting good results with policy gradient methods, which are very sensitive to stepsize and also typically take millions or billions of timesteps to learn simple tasks due to their poor sample efficiency [11].

The objective function maximized in PPO is represented as:

$$L(\theta) = \hat{\mathbb{E}}_t \left[min \left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{4}$$

where θ is the policy parameter, $\hat{\mathbb{E}}_t$ is the empirical expectation over timesteps, r_t is the ratio of the probability under new and old policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a clipping hyperparameter (for which we used the default of 0.2) [11].

PPO makes use of the actor-critic architecture, which consists of (1) an actor, which aims to choose the best action based on a given state and (2) a critic, which produces a Q-value, represented as

$$Q(s, a) = V(s) + A(s, a) \tag{5}$$

where $V(s)$ is the value function given a state and $A(s, a)$ is the advantage function given a state and action. The advantage function reveals how good a particular action is compared to other actions at a given state while the value function describes how good it is to be in a particular state. The actor and critic each improve at their jobs over time to learn the optimal policy and action evaluations, respectively.

In DroneDR, both the policy function and the value function were represented with a Multilayer Perceptron (MLP) policy network comprised of 3 layers of size 128 each. For the PPO implementation, we used the framework provided by Stable Baselines [12].

4 EXPERIMENTAL SETUP

No two post-disaster regions are the same. The size of the affected region, the movement patterns of the ground nodes, the number

of drones available, and the capabilities of those drones (e.g. their radio ranges) are just a few of the factors that can vary drastically across different disaster-struck areas. It is critical for an emergency response network to be high-performing regardless of the specifics of a particular disaster. Thus, we evaluate the robustness and adaptability of our algorithm on a diverse set of inputs by considering two very different types of environments (implemented as custom OpenAI Gym [13] environments and described below). In all environment descriptions, the node range (r_n) is defined based on the x-y coordinates of discretized grids occupied by nodes and drones. In particular, a node at grid U is connected to a drone at grid D (and therefore has a positive connection status) if:

$$\frac{max(|U[x] - D[x]|, |U[y] - D[y]|)}{Grid\_size} \leq 2 \quad (6)$$

i.e, if their respective normalized locations are withing 2 units of each other in the projected x-y coordinate plane.

4.1 Small Disaster-Struck Environment

This particular hypothetical environment assumes that the emergency response crew is concentrated in a small region of 100x100 square meters. The environment represents a smaller-scale disaster or emergency situation, such as a neighborhood fire like the recent Calumet Heights blaze [14].

In this small disaster environment, the ground area and its airspace is discretized into small grid cells of size 5m x 5m. In such a small and focused region, first responders will have specific tasks, whether that is a search-and-rescue mission or providing parademic services to victims. They also will move in groups based on their role and therefore, their movement is modelled by the Reference Point Group Mobility (RPGM) model [15]. The RPGM model was designed for group mobility, in which each group has a designated *leader* node that serves as a reference point for the other nodes in the group. Each non-leader node in the group then deviates from the leader by a random deviation vector. As mentioned in [16–20], the RPGM model is commonly used to model the movement of rescue crews (e.g. police, firefighters, medical assistants) during disaster relief missions, since these crews tend to form groups as they carry out their rescue tasks. To incorporate the RPGM node movements in this environment, we used the pymobility framework [21].

4.2 Large Disaster-Struck Environment

This disaster scenario represents a larger emergency situation, such as the September 11 attacks. To more accurately model a real-world urban environment, we based this large-scale urban environment on a 1 sq km area in downtown Chicago. We constructed a 3D CAD model of buildings in the chosen area using OpenStreetMap [22] and used ray tracing techniques to measure Received Signal Strength (RSS) for many pairs of randomly sampled drone, node coordinates.

To generate RSS measurements, we used the WirelessInsite software [23] and performed radio frequency ray tracing, which considers the 3D environment's effect on radio wave propagation. Specifically, we obtained the received power between different transmitter-receiver pairs at various randomly sampled coordinates in the airspace and on the ground. Figure 2 shows an example

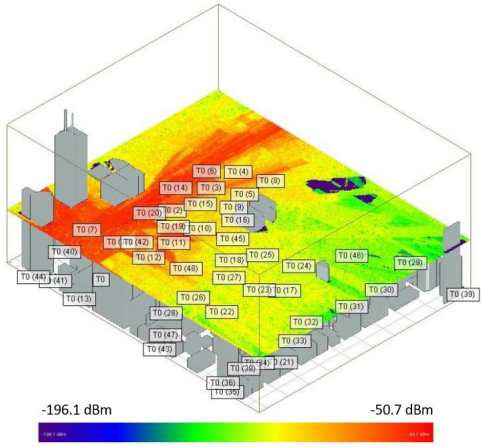

Figure 2: Heatmap representing received power from a transmitter on the ground to grid of points representing drone positions located at 200 m above ground. Note that this heatmap is only for that one transmitter point and there will be a different heatmap for each transmitter point.

heatmap of the RSS from a transmitter on the ground to a grid of receivers in the sky.

The sampled individual transmitter-receiver pairs were grouped into grid cells of size 50m x 50m and an average RSS was extracted for each pair of transmitter-receiver grid cells. This average RSS would then define the expected signal strength for a connection between a drone and node occupying their respective cells.

The RSS heatmap information is used to determine whether there is a connection between a drone and a node at a given point in time. It is used as a lookup table with the key being the pair of (drone, node) positions and the value being the RSS for that pair of positions. Therefore, if either a drone or a node moves to different grid cell, the RSS measurement between that drone and node changes. We define a node as having a positive connection status using the same requirements described in Section 4, with the added condition that the average RSS from the node (in its current grid location) to at least one drone is greater than a given threshold (which we set to -250 dBm).

The movement of user nodes in this case is also assumed to be different from a small scale environment's because first responders are expected to cover greater distances in a larger, more open region. Particularly in a large urban environment, it is possible for first responders to move similarly to troops. In fact, military forces have played a key role in disaster relief situations in the past [24] and prior works have proposed military mobility models for delivering post-disaster critical care services [25]. Therefore, node mobility in this large-scale disaster environment was modelled based on the military mobility model proposed by [26] with the assumption that each node moves independently.

5 EVALUATION

In this section, we present a comparison of DroneDR's performance in the two chosen classes of environments with that of other baseline algorithms. In particular, we demonstrate the effectiveness

of our proposed reinforcement learning approach against the following baselines: (1) Greedy Steiner Heuristic and (2) Random Waypoint movement.

The Greedy Steiner heuristic is based on the NP-hard budget Steiner problem formulated in [27] for general graphs. In the budget Steiner problem, a subset of vertices (dubbed as terminals) are associated with an importance value and the algorithm tries to find the tree with highest cumulative importance connecting these terminals with a limit (or budget) on tree size. It can be applied to our problem by constructing a graph at each timestep with vertices representing all feasible discretized grids that could be occupied by ground nodes and drones. Edges are added between two vertices if they are within radio range. Grids which host ground nodes are designated as terminals and a size constrained Steiner tree interconnecting terminal nodes is determined using the approximation algorithm proposed in [27]. Non-terminal grids (also typically referred to as Steiner points) identified in the resultant Steiner tree inform the desired positions of drones. The implemented heuristic computes desired drone positions given the current state of the system and greedily moves each drone from its current position at each timestep until all drones reach their desired target positions.

The Random Waypoint (RWP) model randomly chooses a direction, speed, and duration for each drone to move in, and once that duration is over, another random set is selected for that drone. In our evaluation, we used the pymobility implementation of the Random Waypoint model [21].

For training, we defined an episode as lasting a length of 300 timesteps, where each timestep represented a discrete interval of 10 seconds. All models were trained for a total of 1 million timesteps (or around 3333 episodes). All experiments were run on a Intel Xeon (E5-2660 v4) server with 64 GB RAM and 56 cores clocked between 2.0-3.2 GHz. After training a RL model for each unique environment condition, we evaluated each model across 100 runs of 300 timesteps each. The main metrics that our evaluation focused on are described as follows:

- *Coverage*: A node is considered "covered" if it has a positive connection status (i.e. it can connect to at least one drone). This metric is calculated per timestep and normalized over the total number of nodes. Coverage is an indicator of the average percentage of nodes with a functioning uplink.
- *Average number of connections*: This metric represents the number of connections that are active (i.e. connections where both of the nodes in the connection pair are connected to at least one drone) and in our configuration of DroneDR, this was the metric that was explicitly optimized. Like the *coverage* metric, this metric should be interpreted on a per timestep basis. The values of this metric can vary from 0 to the total number of desired connections and the values shown in our results were calculated by averaging the number of connections maintained per timestep across all 100 evaluation runs.
- *Fairness*: Fairness is a measure of the algorithm's tendency to provide unbiased service to nodes. To evaluate the fairness of our solution, we used Jain's fairness index, where a value of 1 indicates equal allocation between all the users considered [28]. We only included non-leader nodes in computing this metric to ensure that fairness was only determined across nodes of equal importance. The values we used to calculate this fairness metric was based on the total number of timesteps per run (averaged across all evaluation runs) for which each node was connected to at least one drone. The maximum fairness value of 1 indicates that all non-leader nodes were connected to at least one drone for an equal number of timesteps.
- *Drone-AP Utilization*: This metric represents the percentage of time (on average) that the drones were serving as access points (APs) and therefore directly connected to at least one node. It was calculated by counting the total amount of time (within one evaluation run) each drone was connected to at least one node, taking the average (for each drone) over all of the evaluation runs, and finally taking the average over all the drones.

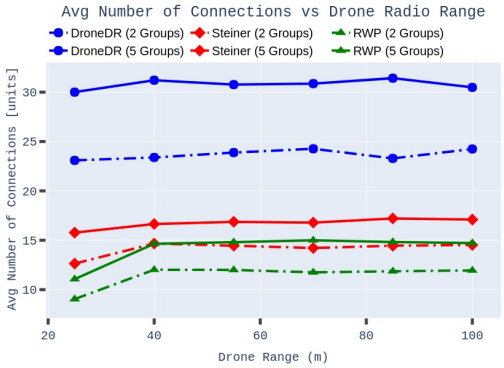

Figure 3: A comparison of average number of connections per timestep for a 10 drone, 40 node, small-scale environment as the inter-drone radio ranges (r_d) are varied.

5.1 Small Disaster-Struck Environment

To simulate realistic emergency response scenarios, we devised experiments involving a fixed number of 40 nodes which were equally divided into a variable number of groups. We evaluated DroneDR's ability to maintain connections when there was a total of 2 groups as well as a total of 5 groups. For each specification, the total number of desired connections accounting for connectivity among group leaders and between each group member and its associated leader evaluates to 39 and 45 respectively.

In Figure 3, the average number of maintained connections is plotted across different drone radio ranges. Increasing the drone radio range relaxes strong connectivity constraints imposed on drone movement and allows the drones to move more freely. From a drone range (r_d) of 40m onwards, the average number of connections is fairly constant for each algorithm. DroneDR consistently outperforms the Steiner heuristic by maintaining over 75% more connections (in the 5-group case) and consistently outperforms RWP by a factor of 2.

The relationship between the average number of maintained connections and the number of available drones is depicted in Figure 4. As expected, all heuristics were able to maintain more connections as the number of available drones was increased. The RL algorithm was able to maintain around 2x more connections on

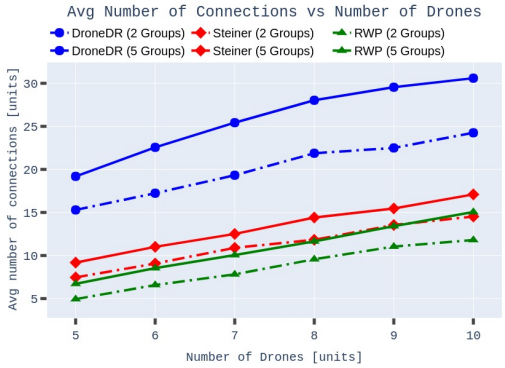

Figure 4: A comparison of average number of active connections per timestep as the number of drones and number of groups are varied in a small-scale environment, while the total number of user nodes and the drone range (r_d) are fixed at 40 nodes and 100m, respectively.

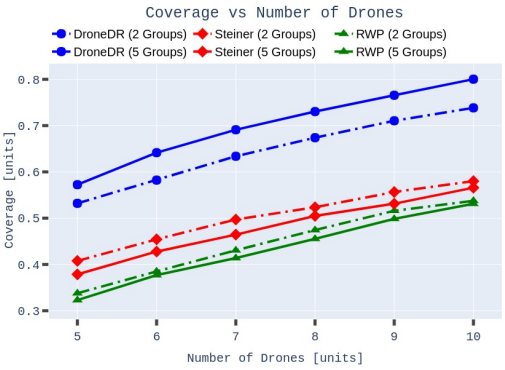

Figure 5: A comparison of coverage per timestep as the number of drones and number of groups are varied in a small-scale environment, while the total number of user nodes and the drone range (r_d) are fixed at 40 nodes and 100m, respectively.

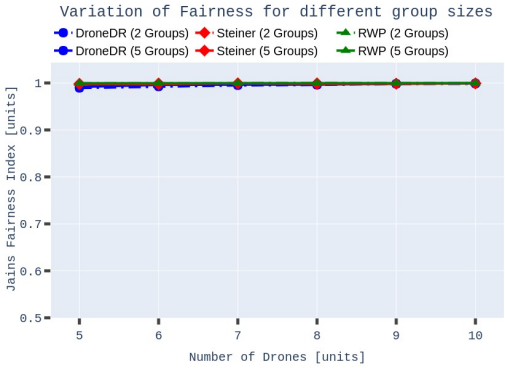

Figure 6: A comparison of Jain's fairness index based on total connectivity period observed by each node as the number of drones and number of groups are varied in a small-scale environment, while the total number of user nodes and the drone range (r_d) are fixed at 40 nodes and 100m, respectively.

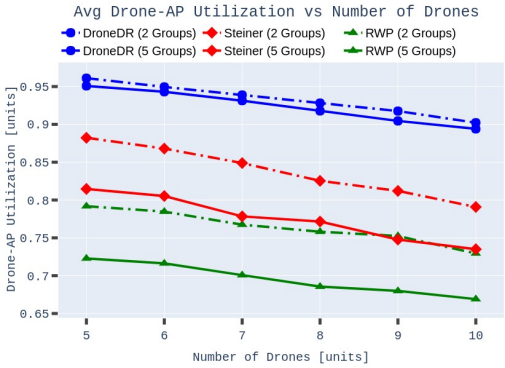

Figure 7: A comparison of the average percentage of time drones were being utilized as APs directly serving user nodes in a small-scale environment where the number of drones and number of groups varied and number of nodes and drone range (r_d) were fixed at 40 nodes and 100m, respectively.

average than the Steiner heuristic (for the 5-group case) and up to 3x more than the average achieved by the RWP heuristic (for the 2-group case).

Figure 5 shows the coverage of the drone network as the number of drones were varied. The RL algorithm consistently covers more nodes regardless of the number of drones available or the number of operating node groups in the environment. For the 2 group case, DroneDR consistently achieves over 10% more coverage than the Steiner heuristic and for the 5 group case, DroneDR consistently achieves almost 20% more coverage than the Steiner heuristic.

A comparison of Fairness score of each algorithm is depicted in Figure 6. The RL algorithm is able to provide unbiased service to all members of all groups.

Figure 7 shows the average drone-AP utilization as the number of drones increased for varying groups and algorithms in a small-scale environment. DroneDR consistently has the highest drone-AP utilization of around 0.9 or above, which means that in general, the drones are constantly positioned at a location that allows them to provide connectivity to user nodes.

5.2 Large Disaster-Struck Environment

Unlike the small scale scenario, in this simulation we experimented with a different mobility pattern where the nodes move independently of each other. In this set of experiments, the total number of nodes in the network was varied. In particular, we considered a scenario involving 40 nodes and a scenario involving 10 nodes. The experiment setup implied an all to one connectivity requirement where one node would be designated as the leader which attempts to connect with all members of the single large group.

In Figure 8, the total number of active connections is plotted as the drone radio-ranges are varied. The greatest difference in the average number of active connections occurred between the change from a 250m drone range to a 400m drone range, and this was true for all three algorithms. DroneDR consistently outperformed RWP

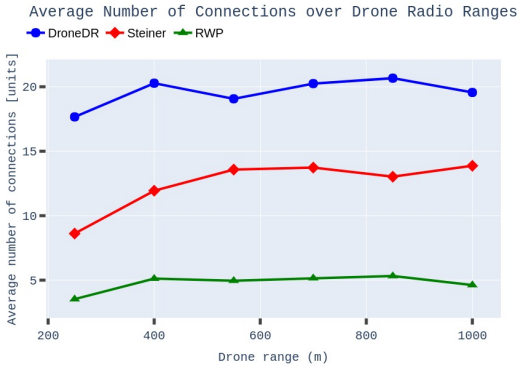

Figure 8: A comparison of average number of connections per timestep for a 10 drone, 40 node, large-scale environment as the inter-drone radio ranges (r_d) are varied.

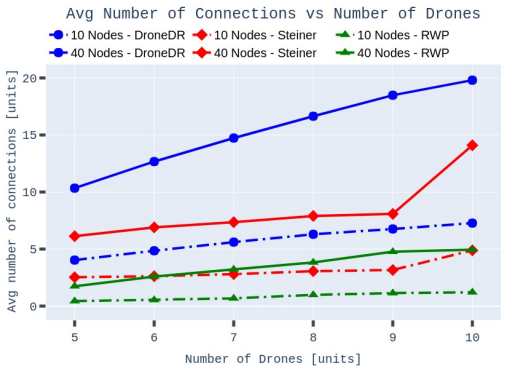

Figure 9: A comparison of average number of connections per timestep for a large-scale environment as the number of drones and number of nodes vary. Drones have a range (r_d) of 1000m.

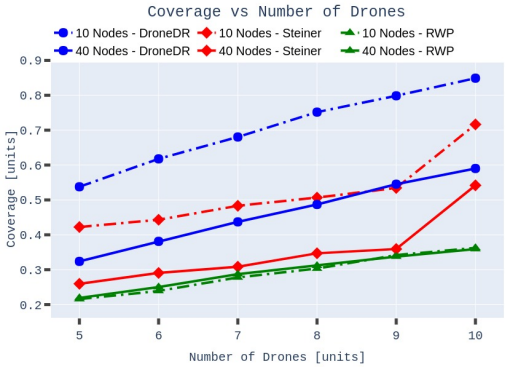

Figure 10: A comparison of average coverage per timestep for a large-scale environment as number of drones and number of nodes vary. Drones have a range (r_d) of 1000m.

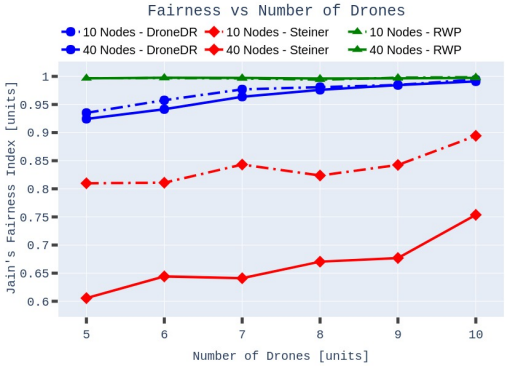

Figure 11: A comparison of Jain's fairness measure, which was computed for the large-scale environment based on total connectivity period observed by each node, as the number of drones and number of nodes are increased while drone range (r_d) is fixed at 1000m.

by a factor of 3.5 or more and also maintained at least 6 more connections on average than the Steiner heuristic across all drone ranges.

In Figure 9, we fixed the drone range (r_d) to 1000m allowing maximum flexibility and varied the number of available drones to measure the average number of maintained connections. In general, there is an increasing trend among the average number of connections for all algorithms as the number of drones is increased. DroneDR still outperforms other baselines, sometimes by more than a factor of 2.

Figure 10 shows the average coverage for each algorithm as the number of drones and nodes are varied. As the number of drones is increased, coverage improves for all algorithms. DroneDR again achieves the highest coverage among the baselines with almost 30% gains for some configurations.

Figure 11 shows the fairness index across different network sizes. The algorithm that performed the worst in terms of average number of active connections and average number of connected nodes (i.e. Random Waypoint) had the best performance in terms of fairness, as can be seen by its near-constant slope close to a fairness value of 1.0. This could be due to the random nature of that baseline algorithm, since unlike the Reinforcement Learning-based and Steiner-based algorithms, Random Waypoint does not consider node roles (i.e. leader or non-leader node) when making drone movement decisions. Even so, DroneDR achieved a good fairness measure of over 0.9 with 5 drones and over 0.95 with 7 or more drones.

Figure 12 shows the average drone-AP utilization as the number of drones increased for a varying number of nodes and various algorithms in a large-scale environment. DroneDR outperformed Steiner (sometimes by a difference of over 0.2) and RWP (sometimes by a difference of over 0.4), reaching over 0.9 drone-AP utilization in the 40 node case and around 0.7 utilization or above in the 10 node case.

6 DISCUSSION

From our extensive experiments, we have shown that DroneDR is able to perform well for a broad spectrum of drone ranges, for a different number of drones and node count combinations, and even for dissimilar environments that differed in the scale, node mobility pattern, and connectivity requirements. Although the reward function used in the training process for DroneDR was based on the

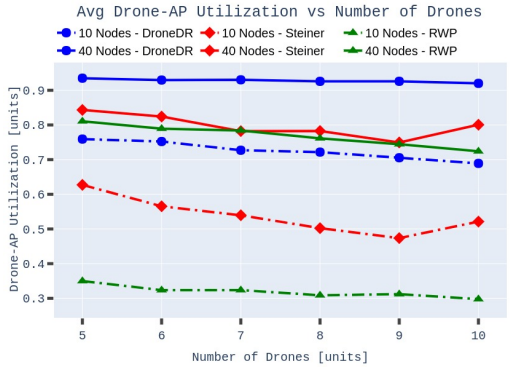

Figure 12: A comparison of the average percentage of time drones were being utilized as APs directly serving user nodes in a large-scale environment where the number of drones and number of nodes varied and drone range (r_d) was fixed at 1000m.

number of active connections, DroneDR still greatly outperformed the non-learning baseline algorithms in other metrics such as coverage and drone-AP utilization. These results are significant because they demonstrate the potential that a RL approach combined with a constrained movement algorithm has for the previously unexplored problem of maintaining node-specific pairwise connectivity requirements in a highly mobile and dynamic network. Our approach also leverages deep neural networks which can be further extended and optimized to handle even larger and more complex environment specifications and connectivity requirements. In the future, we plan to further improve DroneDR in the following ways:

Beyond Connectivity. Our work has thus far focused on maximizing the number of established connections. As a next step, we plan to explicitly incorporate additional requirements, such as fairness and energy constraints into our reinforcement learning approach. This would require modifications to the MDP formulation and environment specification.

Transfer Learning. Another potential area of improvement is to modify our reinforcement learning model to handle a wide array of tasks by applying knowledge learned from one task onto another task. Although our approach performs well for a wide range of environments, a model that was trained in one specific environment could not be directly applied to make actionable decisions even in a slightly altered environment. This means that if there is a sudden change in the environment (such as a drone failure), then a new model would need to be trained for the altered environment. We need a methodology for rapid and robust handling of failure scenarios where the number of available drones and nodes could evolve over time.

A potential solution for this problem could be to leverage model agnostic meta learning (MAML) techniques. MAML methodology involves training a single reinforcement learning agent to function well for a wide range of tasks. It proceeds by finding a common set of policy parameters close to the optimal parameters of each task [29]. Once this policy is found, it can be adapted to different tasks in a small number of steps by observing a small number of samples. In our case, the effects of different types of failures (i.e. the number of functional drones and nodes remaining) can be encoded into separate tasks, so at a high level, tasks differ in the number active nodes in the system. Training a MAML-hardened algorithm can make it resilient to unexpected failures.

Hardware Testbeds. Through simulation-based evaluation, we have shown that our proposed approach outperforms two baselines: Random Waypoint and a Greedy Steiner heuristic across a broad range of environments. Next, we plan to evaluate our approach on a real testbed involving physical drones to observe the effect of real world uncertainties on algorithm performance.

7 RELATED WORK

The problem of optimal drone placement and UAV organization has been widely explored in the past, both in the context of disaster relief and more generally. On the disaster relief side, a UAV-aided emergency network was proposed by [6], which focused on developing a deployment tool for a UAV-aided network that considered drone specifications, user requirements, and the environment's 3D model. In [5], the authors proposed Emergency Wi-Fi Network on Drone (WiND), which used hexagonal cells in a honeycomb pattern as a basis for the drone positioning algorithm. In both [6] and [5], the drones settle into place after reaching their optimal locations, which differs from our approach of constantly re-positioning the drones in response to environment dynamics.

The work in [30] aimed to use UAVs as relays to repair network holes in a post-disaster scenario and did so by formulating the problem into a binary integer program. However, the networks they considered were stationary ad-hoc networks whereas our work focuses on mobile IoT networks. Others works, such as [31], used machine learning methods (in particular, Q-learning) for finding the best position for drones in an emergency scenario. While we also take a RL-based approach, our work differs in the underlying assumptions we make (some nodes are more important than others, drones need to be fully connected, etc.), the applications (diverse types of environments), and the main objectives (maximum number of connection pairs, not just total number of users covered).

More generally, past works have used methods such as particle optimization [32, 33], bare bones fireworks algorithm [34], circle packing theory [35], and RF ray tracing and real-world measurements [36, 37] to position UAVs so that some objective is met (whether that is maximizing coverage area [35], optimizing for modified global message connectivity [33], maximizing SNR to all users [36], or maximizing user coverage [32, 34, 37]). These differ from our objective of maximizing the number of node-to-node connections satisfied.

Some works, such as [38], had the added constraint of maintaining connectivity amongst all the UAVs in the network. However, while their focus was on either using the minimal number of UAVs to satisfy service requirements or providing optimal coverage given a number of available UAVs, our main objective was to maximize the number of active connections, which also considered different node roles (i.e. leader or non-leader). Furthermore, a key contribution of our work was the addition of reinforcement learning to this problem domain, which allowed our algorithm to make decisions based on not only the current state, but also previous and predicted future states.

8 CONCLUSION

In this paper, we developed DroneDR, a framework which uses reinforcement learning to intelligently position UAVs serving as airborne relays for user nodes. DroneDR continuously observes current node locations at discrete timesteps and strives to position UAVs to maximize the number of node-to-node connections while maintaining a strongly connected drone network. The proposed approach combines the benefits of intelligence gained from the learning-based approach for selecting which nodes to prioritize, with the safety provided by the more restrictive drone movement algorithm.

DroneDR was evaluated on two different emergency response scenarios in (1) a small-scale post-disaster environment and (2) a large-scale disaster-struck environment. For all considered environment specifications, we show that our algorithm achieved significant performance improvements over other baseline techniques. DroneDR achieved up to around 2x the average number of connections achieved by the Steiner greedy heuristic algorithm, and over 3x the amount achieved by the Random Waypoint model. It was also able to maintain a high amount of coverage, with up to 30% increase in coverage compared to the the Steiner algorithm. In addition, DroneDR provides quality communications to all nodes during the event, achieving a Jain's fairness index score of over 0.9 in every scenario. We believe these results provide some demonstration of the benefits of using reinforcement learning techniques to aid communication during disaster relief operations.

ACKNOWLEDGMENT

This work was supported by Boeing Research & Technology (BR&T) under the Collaborative Research Project BRT-Z0518-5050: Deep Learning-based Tactical IoT Networking in Contested Environments. The authors would like to thank Dr. Jae H. Kim (Boeing PM) for his advice and guidance throughout the project.

REFERENCES

[1] Amnesty International. Cyclone Idai: One month after devastating cyclone, more international assistance needed to protect people's rights, 2019. Accessed: 2019-12-22.
[2] AFP-JIJI. Tourists stranded and at least 16 dead after Typhoon Phanfone in Philippines, 2019. https://www.japantimes.co.jp/news/2019/12/26/asia-pacific/typhoon-phanfone-philippines/.
[3] G. T. C. Gunaratna, Pahan Jayarathna, S. S. P. Sandamini, and D. S. De Silva. Implementing wireless adhoc networks for disaster relief communication. *2015 8th International Conference on Ubi-Media Computing (UMEDIA)*, pages 66–71, 2015.
[4] Paul Bupe, Rami J. Haddad, and Fernando Rios-Gutierrez. Relief and emergency communication network based on an autonomous decentralized UAV clustering network. *SoutheastCon 2015*, pages 1–8, 2015.
[5] Kirtan Gopal Panda, Shrayan Das, Debarati Sen, and Wasim Arif. Design and deployment of UAV-aided post-disaster emergency network. *IEEE Access*, 7:102985–102999, 2019.
[6] Margot Deruyck, Jorg Wyckmans, Wout Joseph, and Luc Martens. Designing UAV-aided emergency networks for large-scale disaster scenarios. *EURASIP Journal on Wireless Communications and Networking*, 2018:1–12, 2018.
[7] Silvia Krug, Matias Siracusa, Sebastian Schellenberg, Peggy Begerow, Jochen Seitz, Thomas Finke, and Juergen Schroeder. Movement patterns for mobile networks in disaster scenarios. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, 06 2014.
[8] Michelle Royal. Project Responder 5. Technical report, Falls Church, Virginia, 2017.
[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[10] PPO1 - stable baselines, 2018-2019. https://stable-baselines.readthedocs.io/en/master/modules/ppo1.html.
[11] John Schulman, Oleg Klimov, Filip Wolski, Prafulla Dhariwal, and Alec Radford. Proximal policy optimization, July 2017. https://openai.com/blog/openai-baselines-ppo/.
[12] Stable baselines - rl baselines made easy, 2018-2019. https://stable-baselines.readthedocs.io/en/master/index.html.
[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016. https://gym.openai.com/.
[14] Deanese Williams-Harris. Firefighter, 3 trapped residents injured in calumet heights neighborhood blaze. *Chicago Tribune*, Jan 2020.
[15] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems - MSWiM 99*, 1999.
[16] Fan Bai and Ahmed Helmy. *A Survey of Mobility Models in Wireless Adhoc Networks (Chapter 1)*, page 1–30. Kluwer Academic, 2004.
[17] Onye Erim and Collin Wright. Optimized mobility models for disaster recovery using UAVs. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, 2017.
[18] Feng Geng and Shengjun Xue. A comparative study of mobility models in the performance evaluation of MCL. *2013 22nd Wireless and Optical Communication Conference*, pages 288–292, 2013.
[19] Deepak Kumar, Ashutosh Srivastava, and Suresh C. Gupta. Routing in ad hoc networks under reference point group mobility. *2013 European Modelling Symposium*, 2013.
[20] Geetha Jayakumar and Gopinath Ganapathi. Reference point group mobility and random waypoint models in performance evaluation of manet routing protocols. *Journal Comp. Netw. and Communic.*, 2008:860364:1–860364:10, 2008.
[21] André Panisson. pymobility - python implementation of mobility models, 2014.
[22] OpenStreetMap. https://www.openstreetmap.org/.
[23] WirelessInsite. http://www.remcom.com/.
[24] Peter Walker. Foreign military resources for disaster relief: An NGO perspective. *Disasters*, 16, 1992.
[25] Jack Sariego. CCATT: A military model for civilian disaster management. *Disaster Management Response*, 4:114–117, 2006.
[26] Kim S. Seo, H. and J. Ma. A novel mobility model for the military operations with real traces. *The 12th International Conference on Advanced Communication Technology (ICACT)*, 2:129–133, 2010.
[27] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: Theory and practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, page 760–769, USA, 2000. Society for Industrial and Applied Mathematics.
[28] Rajendra Jain, Dah-Ming Chiu, and William Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *DEC-TR-301 Tech. Rep.*, 1984.
[29] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference on Machine Learning (JMLR)*, 70, 2017.
[30] Dahee Jeong, So-Yeon Park, and HyungJune Lee. DroneNet: Network reconstruction through sparse connectivity probing using distributed UAVs. *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1797–1802, 2015.
[31] Paulo V. Klaine, João P. B. Nadas, Richard D. Souza, and Muhammad A. Imran. Distributed drone base station positioning for emergency cellular networks using reinforcement learning. *Cognitive Computation*, 10(5):790–804, 2018.
[32] Elham Kalantari, Halim Yanikomeroglu, and Abbas Yongacoglu. On the number and 3D placement of drone base stations in wireless cellular networks. *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, 2016.
[33] Pawel Ladosz, Hyondong Oh, and Wen-Hua Chen. Optimal positioning of communication relay unmanned aerial vehicles in urban environments. *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1140–1147, 2016.
[34] Eva Tuba, Ira Tuba, Diana Dolicanin-Djekic, Adis Alihodzic, and Milan Tuba. Efficient drone placement for wireless sensor networks coverage by bare bones fireworks algorithm. *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018.
[35] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Merouane Debbah. Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage. *IEEE Communications Letters*, 20(8):1647–1650, 2016.
[36] Ashutosh Dhekne, Mahanth Gowda, and Romit Roy Choudhury. Extending cell tower coverage through drones. *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications - HotMobile 17*, 2017.
[37] Ayon Chakraborty, Eugene Chai, Karthikeyan Sundaresan, Amir Khojastepour, and Sampath Rangarajan. Skyran. *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies - CoNEXT 18*, 2018.
[38] Haitao Zhao, Haijun Wang, Weiyu Wu, and Jibo Wei. Deployment algorithms for UAV airborne networks toward on-demand coverage. *IEEE Journal on Selected Areas in Communications*, 36:2015–2031, 2018.