An Analysis of Robustness for Multi-Agent Reinforcement Learning in UAV Field Coverage

Evan Czyzycki
Department of Computer Science
University of California Los Angeles
Los Angeles, CA, 90095
`eczy@cs.ucla.edu`

Abstract

This study explores the robustness of a previously proposed Multi-Agent Reinforcement Learning (MARL) algorithm which addresses the problem of field coverage by a team of unmanned aerial vehicles (UAV). This algorithm allows a team of UAV equipped with ground-facing cameras to cooperatively learn to provide full view of an unknown field of interest while minimizing the overlap among their camera views. The evaluation of this method in the original study is performed only on a static environment. In order to determine the performance of this approach on more dynamic environments, this study evaluates the proposed algorithm on environments where the field of interest, camera field of view, or both are dynamically modified in the environment. Additionally, this study includes published code, which the original study fails to provide.

1 Contributions

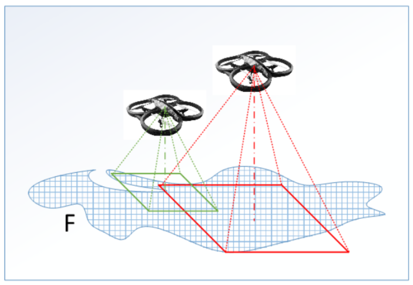
Yuxuan Chen (yuxuan7@g.ucla.edu) assisted with the original project proposal and the project checkin report. All code implementation and analysis as well as the poster, the video, and this final report was completed by Evan Czyzycki (contact information above).

2 Introduction

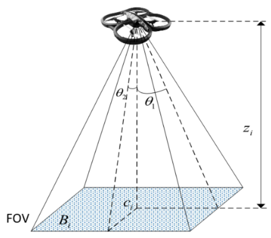
Optimal sensor coverage in general is the problem of maximizing the effectiveness of a network of sensors collecting data in often complex environments. This is an active field of research with many proposed solutions to many specific instances of this problem [5]. However, many of these solutions rely on underlying assumptions regarding the nature of the environment. These assumptions are often represented as environment models which become exponentially more difficult to design as the number of agents increases or the complexity of the system increases [5]. By framing these problems as reinforcement learning problems, we are able to approach them with algorithms that do not rely on predefined models of the environment but rather allow agents to learn about optimal behavior and the nature of the environment independently.

The specific problem we examine in this study is the problem of optimal field coverage using a swarm of UAV drones. In this problem, it is of interest to use the swarm as a network of sensors to obtain visual information of a particular area of interest on the ground, e.g. a wildfire or an oil spill [5]. A study conducted by Pham et al. [5] proposes a distributed, cooperative algorithm which is shown to optimally cover a defined field of interest with minimal overlap in drone sensors in a simple discrete simulation. However, in their study, the environment is static. For every episode, each drone maintains the same field of vision, the field of interest remains the same, and the number of drones does not change. In more realistic environments, these properties may not hold. A wildfire

Preprint. Under review.



(a) Depiction of an environment with two drones hovering over an unknown field of vision. Source: Pham et al. [5].



(b) Drone field of view representation. In our implementation, $\theta_1 = \theta_2$ for all drones in all environments such that every drone has a square FOV projection on the ground. Source: Pham et al. [5].

Figure 1

for example will very likely not remain in the same shape and location, and drones may be added or removed at any time. In our study, we evaluate the robustness of the solution proposed in [5] for environments where drone field of view modified, where the field of interest is modified, and where both drone field of view and the environment field of interest is modified.

3 Problem Formulation

In our field coverage problem, a swarm of UAV drones are sent out as a sensor network with the goal of collecting as much visual information as possible about a particular field of interest (FOI) which lies on the ground. Each drone behaves as an independent agent and is able to move in 3D space, locate itself via GPS, communicate with other drones, and collect visual information through a ground-facing camera. The goal of the UAV swarm is to maximize the FOI area covered by all drone cameras while simultaneously minimizing the overlap in field of view (FOV) among all drone cameras in order to maximize the resolution of the imaging. This is shown in Figure 1a.

In our specific task, the camera attached to each drone has a pyramidal FOV with a predefined angle which determines the projected square area on the ground such as in Figure 1b. The field of vision to be covered is completely flat and uniformly weighted. This problem formulation is identical to that defined in Pham et al. [5].

4 Methods

Our methodology is very similar to that in Pham et al. [5] with the addition of dynamic environments. However, since this study did not publish any code, we implemented a custom simulator for this task in addition to our agent based on the Multi-Agent Approximated Equilibrium-Based Q-Learning algorithm defined in Pham et al. [5]. This algorithm is rewritten with some of our particular implementation details in Algorithm 1.

4.1 Simulator

Our simulator is implemented as an OpenAI Gym [1] environment. The map is represented as a three-dimensional NumPy [2] array of size (X, Y, Z). The ground on which the FOI lies is the slice of this array where $z = 0$. The field of interest is represented as a set of points on this plane, and each drone is represented as a single point in the three-dimensional space above the ground ($z > 0$).

Every drone in the environment has the same square pyramidal FOV, characterized by its half-angle θ. At each time step, every drone may move one discrete unit in one of six directions: up, down, right, left, forward, backward. Up and down movement is defined as positive and negative movement along

[1]`https://gym.openai.com/`
[2]`https://numpy.org/`

the Z-axis, right and left along the X-axis, and forward and backward along the Y-axis. These actions are taken jointly at each time step. When these actions are taken, the drones follow a priority order in order to prevent collisions. When executing the joint action, if two drones will transition to the same state after taking their individual actions, the drone with higher priority will take action and the drone with lower priority will not move. This is an example of *social conventions* [1] in multi-agent reinforcement learning.

The reward function is defined as a global reward over all drones in the environment. At each timestep, each agent receives a reward if and only if the overall goal is acheived:

$$GR(S_t, A_t) = \begin{cases} r & if \sum_i f_i(S_t) \geq fb, \sum_i o_i(S_t) \leq 0, \forall i \in 1, ..., n \\ 0 & otherwise \end{cases} \tag{1}$$

where $f_i(s)$ is the area of the field covered by drone i in state s, $o_i(s)$ is the overlap of the coverage of drone i in state s with all other drones, and $fb \in \mathbb{R}$ is a predefined lower bound on the desired amount of FOI coverage. In our simulator, $r = 0.1$ and fb is equal to the total area of the FOI. This way, each drone observes a reward of 0.1 if the FOI is fully covered with no overlap, and otherwise each drone observes a reward of 0.

In each episode (i.e. every time the environment is reset), the FOV, FOI, and number of drones remains static unless explicitly modified. The only varying factor across episodes under normal circumstances is the initial state of the drones. All of these properties are identical to the simulator described in Pham et al. [5]. In our implementation, we add functionality which allows for modifications of the environment during training so that we can observe how they might affect agent performance.

4.2 Algorithm

We follow the algorithm proposed in Pham et al. [5] as shown in Algorithm 1. In order to reduce the massive state space of this multi-agent problem at high numbers of drones, the algorithm relies upon approximation techniques when computing state-action values. Pham et al. [5] explore using both Fixed Sparse Representation (FSR) and Radial Basis Function (RBF) techniques to accomplish this [2], and they found in their experiments that FSR consistently achieved higher performance. Thus, in our experiments we only use FSR as our approximation scheme.

Under FSR, we approximate the tabular $Q(S, A)$ function using a parameter vector θ (unique to each agent) and a basis function $\phi : \{S\} \times \{A\} \to \mathbb{R}$. In the same style described in Geramifard et al. [2], we define a state basis function $\phi(s)$:

$$\begin{gathered} \phi(s) = [\phi_{11}(s) \cdots \phi_{1n_1}(s), \phi_{21}(s) \cdots \phi_{2n_2}(s), \cdots, \phi_{d1}(s) \cdots \phi_{dn_d}(s)]^T, \\ \phi_{ij} = \begin{cases} 1 & s^i = v_i^j \\ 0 & \text{otherwise} \end{cases}, i = 1, \cdots, d, j = 1, \cdots, n_i \end{gathered} \tag{2}$$

This is then extended to a state-action basis function $\phi(s, a)$:

$$\phi(s, a) = [\phi(s)[a = A_0], \phi(s)[a = A_1], \cdots, \phi(s)[a = A_n]]^T \tag{3}$$

where $[a = A_i]$ is an indicator function denoting action a is equal to action $A_i \in \{A\}$. I.e. $\phi(s, a)$ is $|\{A\}|$ stacked vectors of $\phi(s)$ where only the indices corresponding to action a can be nonzero. This allows us to learn distinct parameter values for each combination of action and approximated state, effectively allowing us to approximate the state-action value function at a much lower dimensionality.

Using our parameter vector θ_i and our basis function $\phi(s, a)$, we can approximate the state-action value function for state S and action A at time t for drone i as:

$$\hat{Q}_{i,t}(S_t, A_t) = \sum_i \phi(S_t, A_t)^T \theta_i \tag{4}$$

In order to promote early exploration and prevent over-exploration after many episodes, the proposed algorithm uses an ϵ-greedy policy with varying values of ϵ. Though Pham et al. [5] state that this approach is used, the details are not reported. Our implementation uses exponential decay from a predefined ϵ_{max} to a predefined ϵ_{min} as follows:

$$\epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min})e^{\frac{-k}{\lambda}} \tag{5}$$

where ϵ_t is the epsilon at time step t, k is the current number of steps in training (over all episodes), and λ is the decay parameter.

When calculating the optimal joint-action at a given timestep, the algorithm evaluates the Correlated Equilibrium (CE) [3] of actions over all drones using each drone's estimate of state-action values. This can be calculated using linear programming (LP) [4]. We use the same LP defined in Pham et al. [5] with only minor notational differences:

$$\pi(S_t) = \underset{A_t \in \{A\}}{\arg\max}\, f(S_t, A_t) \tag{6}$$

where $f(S_t, A_t)$ is the solution to the linear program

$$\begin{aligned}
&\text{maximize} && \textstyle\sum_{i=0}^{n} Q_{i,t}(S_t, A_t)P_i(a_t)\\
&\text{subject to} && \textstyle\sum_{a_t \in A_{i,t}} P_i(a_t) = 1, \forall i = 1, ..., n\\
& && P_i(a_t) \geq 0, \forall i = 1, ..., n, \forall a_t \in A_t\\
& && \textstyle\sum_{a'_t \in A_{i,t}} [Q_{i,t}(S_t, a_t, A_{t,-i}) - Q_{i,t}(S_t, a'_t, A_{t,-i})]P_i(a_t)\\
& && \geq 0, \forall i \in 1, ..., n
\end{aligned}$$

where $P_i(a_t)$ is the probability of drone i taking individual action a at time t, and A_{-i} represents actions of all drones other than i. I.e. $Q_{i,t}(S_t, a'_t, A_{t,-i})$ represents the Q-value for drone i at time t if drone i takes action a' and all other drones take their respective action from joint-action A_t.

4.3 Experiments

When training the drones in our simulator, we observe the convergence of the proposed algorithm under the following environment dynamics: baseline (static environment as in Pham et al. [5]), changing field of view, changing field of interest, changing both field of view and field of interest.

In all of our experiments, we use an environment of shape $(7, 7, 4)$ containing 2 drones. These values were chosen to be as similar to the procedure in Pham et al. [5] as possible within the bounds of our computational resources. The half-angle of the FOV of each drone is 30°, and the default FOI is the FOI depicted in Figure 2a. These properties remain static unless otherwise stated. We use the learning parameters $\gamma = 0.9$, $\alpha = 0.1$, $\epsilon_{max} = 0.95$, $\epsilon_{min} = 0.05$, $\lambda = 10000$, and $k = 2000$. The duration of each experiment is 500 episodes with any modifications to the environment occurring at episode 250. The maximum number of steps for any single episode is 2000 steps.

Our experiments are as follows:

1. Baseline: drone FOV and FOI remain static.
2. Modifying FOI: FOI is translated 2 units left as shown in Figure 2b at episode 250.
3. Modifying FOV: FOV is widened to $45°$ at episode 250.
4. Modifing FOI and FOV: FOV is widened to $45°$ and FOI is replaced with alternate FOI 1 depicted in Figure 2c at episode 250.
5. Modifing FOI and FOV: FOV is widened to $45°$ and FOI is replaced with alternate FOI 2 depicted in Figure 2d at episode 250.

We only explore a widening FOV because narrowing the FOV may render the FOI unable to be covered, and therefore the algorithm will be unable to converge. We perform experiments with both alternate FOI 1 and alternate FOI 2 in order to explore different FOI with optimal states both similar (alternate FOI 1) and dissimilar (alternate FOI 2) to the original FOI.

Algorithm 1: Multi-Agent Approximated Equilibrium-Based Q-Learning

```
Input  : Parameters: Discount factor $\gamma$, learning rate $\alpha$, maximum epsilon $\epsilon_{max}$, minimum
         epsilon $\epsilon_{min}$, epsilon decay rate $\lambda$, maximum steps per episode $k$
Input  : Basis Function: $\phi(S, A)$
Initialize $\theta_{i,0} \leftarrow 0, \forall i = 1, ..., n$
Initialize $\epsilon_0 = \epsilon_{max}$
Initialize $steps = 0$
for $episode = 0, 1, ...$ do
    Randomly initialize $S \in \{S\}$.
    for $t = 0, 1, ..., k$ do
        $\epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min})e^{\frac{-steps}{\lambda}}$
        for $i = 0, 1, ..., n$ do
            Obtain states $s_{j,t}$ and parameters $\theta_{j,t}$ from other drones $j = 0, 1, ..., n, i \neq j$
            $\pi(S_t) = \begin{cases} \text{Optimal joint-action from solving ...: } A_t & \text{with probability 1 - } \epsilon_t \\ \text{Random joint-action: } A_t & \text{with probability } \epsilon_t \end{cases}$
            Take action $A_t = \pi(S_t)$ according to social conventions rule.
            Obtain states $s_{j,t+1}$ from other drones $j = 0, 1, ..., n, i \neq j$
            $r_t = GR(S_t, A_t)$
            $\theta_{i,t+1} = \theta_{i,t} + \alpha[r_t + \gamma \max_{A' \in \{A\}}(\phi^T(S_{t+1}, A')\theta_{i,t}) - (\phi^T(S_t, A_t)\theta_{i,t}]\phi(S_t, A_t)$
        end
        $steps \leftarrow steps + 1$
    end
end
Output : Parameter vectors $\theta_i, \forall i = 1, ..., n$, joint state-action policy $\pi$
```

Multi-Agent Approximated Equilibrium-Based Q-Learning from Pham et al. [5] with minor implementation modifications.

5 Results

In our baseline experiment, we are able to successfully reproduce the findings of Pham et al. [5]. In Figure 3 we see convergence identical to the results of Pham et al. [5]. Episode length is initially large as the agent explores the environment and converges as the number of episodes increases. After approximately 100 episodes the agent is able to solve the environment in a very low number of steps.

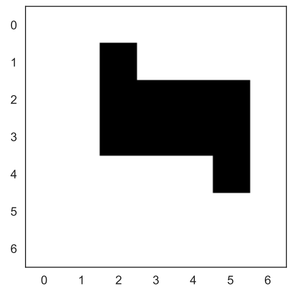
When modifying the FOI by transforming it two units to the left, we see a very similar pattern for the first 249 episodes. In Figure 4 the agent initially takes a large number of steps to solve the environment but eventually converges after around 100 episodes such that it can solve the environment with very few steps. Once the FOI is modified at episode 250, however, we see a small spike in episode length as the agent re-learns to cover the new FOI. After approximately 50 more episodes the agent again converges such that it is able to cover the FOI in a small number of steps.

Increasing the agent FOV to $45°$ does not appear to affect the convergence of the algorithm. As shown in Figure 5, we do not observe a spike in episode length at episode 250 as we do when modifying the FOI. Similarly, increasing the agent FOV to $45°$ and swapping the FOI to the alternate FOI 1 shown in Figure 2c does not appear to affect algorithm convergence. Figure 6 shows a similar pattern where there is no discernible change in episode length at episode 250.

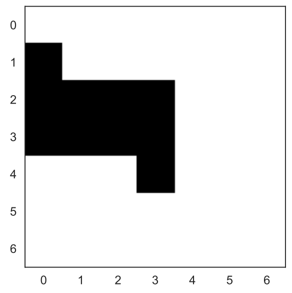
We observe the largest effect in algorithm convergence when increasing agent FOV to $45°$ and swapping the FOI to the alternate FOI 2 shown in Figure 2d. As shown in figure 7. at episode 250 we observe a very large spike in episode length. After approximately 50 additional episodes, however, the agent again is able to solve the environment in few steps.

6 Discussion

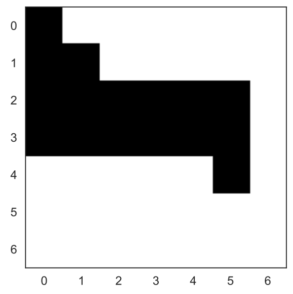
Our results show that the proposed algorithm in Pham et al. [5] is robust to changes in field of view, field of interest, and both field view and field of interest simultaneously within the relatively small



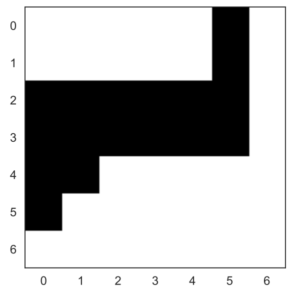
(a) Original static FOI



(b) Original FOI transformed left two units



(c) Alternate FOI 1



(d) Alternate FOI 2

Figure 2: Fields of interest used in FOI environment dynamics.

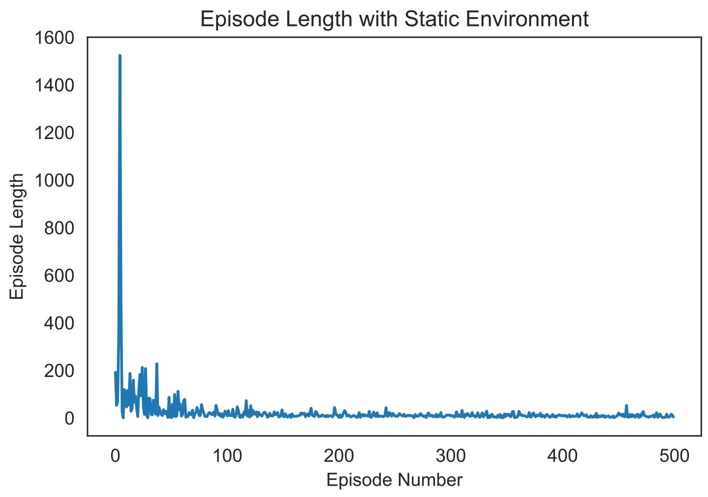


Figure 3: Caption

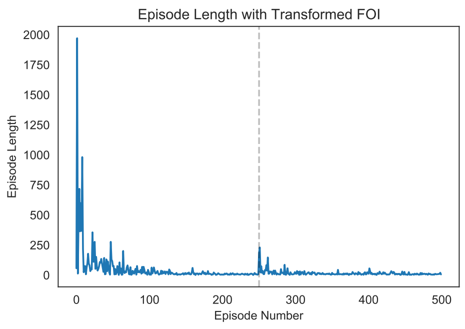


Figure 4: Caption

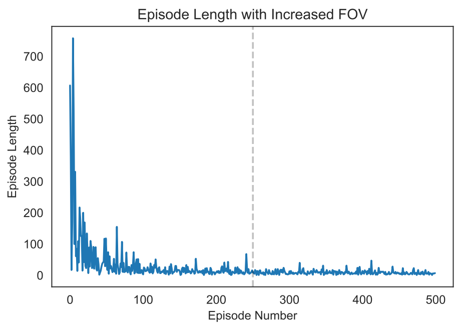


Figure 5: Caption

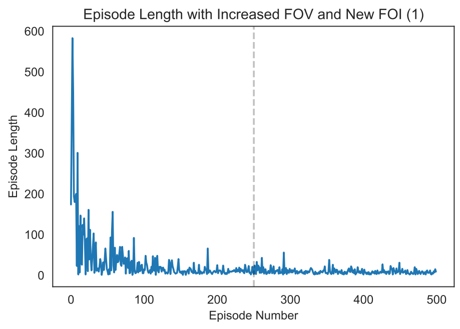


Figure 6: Caption

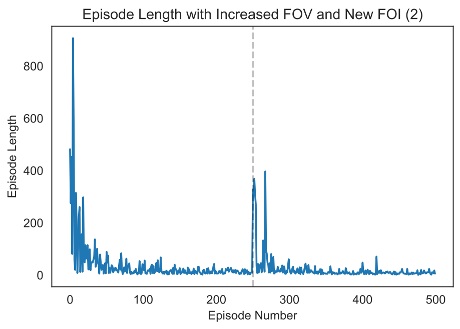


Figure 7: Caption

environment within which we conducted our experiments. When perturbing the environment in these ways, we observe either no effect (experiments 3 and 4) or an increase in episode length quickly followed by re-convergence (experiments 2 and 5). In none of our experiments was the agent unable to converge to the optimal solution after environment modifications.

We suspect, however, that this algorithm is somewhat unstable. The global reward function defined in Equation 1 is sparse, and if the agent is unable to find the solution state randomly within the first few episodes, the agent may not be able to efficiently converge due to decreasing likelihood of exploration. Additionally, this property may cause the algorithm to be less robust to environment changes in larger state spaces. The alternate FOI we used in experiments 4 and 5 are different, but still similar, to the original FOI. In a larger map, an alternate FOI sufficiently different (i.e. translated far enough away) from the original FOI may cause the agent to very slowly re-converge due to its decayed exploration never bringing it close to the new optimal state. The agent would instead rely upon the random initialization to bring it close to the new FOI, which may be very unlikely depending on the size of the environment. Perhaps a less sparse reward function would allow the algorithm to behave more robustly in such environments. Further experimentation is needed to test these hypotheses.

The greatest limit to the practicality of this algorithm is that it relies upon the environment being static across all episodes. The trained agent is therefore only useful for the specific field of interest on which it was trained. A much more useful, albeit more difficult, algorithm would discover the optimal coverage of an unknown FOI which is randomly initialized with each episode. Additionally, the representations and methods in Pham et al. [5] rely upon a fixed number of drones and a fixed environment size. Again, a more useful, but more difficult, distributed algorithm would be able to discover the optimal coverage of an unknown FOI without fixing the number of drones and the size of the environment. Perhaps this algorithm could be extended to handle such cases by copying state-action estimates (i.e. $\phi(s,a)$) into a larger representation vector upon the discovery of new states or upon the addition/removal of drones.

7 Conclusion

In this study, we have implemented the distributed, cooperative multi-agent reinforcement learning algorithm for UAV field coverage proposed in Pham et al. [5] and evaluated its robustness in environments where the drone field of view, the field of interest, or both are not static. We have found that the algorithm, at least in the relatively small environment used both in the original study and in ours, is robust to such environments.

Some interesting future directions to this work which we plan to explore include:

1. Exploration of representations and methods that allow learning of **arbitrary** fields of interest (i.e. cases where FOI is not identical across episodes as in this study).
2. Usage of representations and methods that do not rely upon fixed number of drones and fixed environment size.
3. Integration with deep reinforcement learning techniques.
4. Replacement of discrete state and action spaces with continuous state and action spaces.
5. Coverage of non-uniformly weighted fields of interest and three-dimensional fields of interest.
6. Performance fields of interest which are not contiguous.

Our code has been submitted along with this report. It has also been made public at `https://github.com/eczy/rl-drone-coverage`. The code on GitHub will be under active development, and so for cleaner, up-to-date code please use the GitHub link.

References

[1] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[2] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. P. How. A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Found. Trends Mach. Learn.*, 6(4):375–451, Dec. 2013. ISSN 1935-8237. doi: 10.1561/2200000042. URL `https://doi.org/10.1561/2200000042`.

[3] Greenwald, Amy and Hall, Keith. Correlated-q learning. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 242–249. AAAI Press, 2003. ISBN 1577351894.

[4] C. H. Papadimitriou and T. Roughgarden. Computing correlated equilibria in multi-player games. *J. ACM*, 55(3), Aug. 2008. ISSN 0004-5411. doi: 10.1145/1379759.1379762. URL `https://doi.org/10.1145/1379759.1379762`.

[5] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian. Cooperative and distributed reinforcement learning of drones for field coverage, 2018.