GALOPP: Multi-Agent Deep Reinforcement Learning For Persistent Monitoring With Localization Constraints

Anonymous Author(s)

Abstract: In this paper, we consider a persistent monitoring (PM) problem where a team of agents need to persistently monitor a dynamic area under localization and communication constraints. We assume the agents have limited communication range and a subset of the agents (anchors) have accurate localization capability, while the rest of the agents (auxiliary agents) have lower localization accuracy. To accurately localize itself, the auxiliary agents must be within the communication range of the anchor agents, directly or indirectly. The agents have to cooperate with each to persistently monitor the complete area by communicating their current state to the other agents. Designing strategies for the agents is highly challenging due to the conflicting objectives of complete coverage and the cooperation for communication. Hence, there is a need to design strategies for agents that can learn to navigate satisfying the objectives from the dynamic environment. We present a Multi-Agent Deep Reinforcement Learning based approach GALOPP, which incorporates localization and communication range constraints along with PM to satisfy the conflicting objectives. The GALOPP architecture consists of a Convolutional Neural Network (CNN) – to generate lower dimensional embeddings from the observations, Graph Attention Network (GAT) – to relay the information of neglected region and agent’s state to other agents, and a Proximal Policy Optimization (PPO) based actor-critic to learn the optimal policies. The results show that using GALOPP, the agents learn a stable policy to persistently monitor the environment in complex dynamic environment. The results also show that GALOPP outperforms greedy and Random baselines approaches.

Keywords: Multi-agent deep reinforcement learning, Persistent monitoring, Cooperative localization, Graph attention networks

1 Introduction

Visibility-based Persistent Monitoring (PM) problem involves the continuous surveillance of a bounded environment by a system of robots [1, 2, 3, 4, 5, 6]. It is of practical importance in several domains due to the ever increasing demands for automation in security. We study the problem of planning trajectories for each agent in a multi-robot system for persistently monitoring a dynamic (stochastic) environment. When the environment becomes increasingly complex and is dynamically changing, it becomes challenging to monitor using any deterministic coordinating strategies for PM [6]. Therefore, there is a need for developing strategies for the agents that can learn on how to navigate from the dynamic environment. One such approach is to use Multi-Agent Deep Reinforcement Learning (MADRL) algorithms to determine the policies for the individual agents to navigate in the environment [7].

In this paper, we consider a scenario where a team of multiple robots equipped with a limited field-of-view (FOV) sensor are deployed to monitor a dynamic environment as shown in Figure 1. We assume the environment does not support GPS. In this case, one can deploy agents with high accuracy IMU (like Honeywell HG1700 [8]) that can enable the agents to accurately localize. However, the cost of high accuracy IMU is in tens of thousands of dollars and hence the operation becomes
highly expensive. On the other hand, we can use low cost IMU for the agents, however they are subject to drift resulting in poor localization. Therefore, in this article, we consider two types of localization agents in the team – anchor and auxiliary agents. The anchor agents have high accuracy Inertial Measurement Unit (IMU) and hence can localize accurately by themselves. However, auxiliary agents have low accuracy IMU and require periodic measurements from the anchor agents to update their position and also share the map. The auxiliary agents use the notion of cooperative localization [9, 10, 11] to localize by communicating with the anchor agents directly or indirectly through other auxiliary agents. Thus, the auxiliary agents will have uncertainty in their positional beliefs. Further, the agents have finite communication range which constrains the auxiliary agents motion to be within the communication range of the anchor agents. In this case, the agents may not be able to monitor the complete region as any communication disconnection from the anchor agents will result in poor localization and hence affects the coverage accuracy. This conflicting objective of monitoring the complete area while ensuring the agents are not detached from the anchor agents makes the problem of determining persistent monitoring strategies for the agents challenging.

Figure 1: Persistent monitoring problem using heterogeneous aerial vehicles. The agents have a limited visibility range and communication range. The anchor agents are equipped high precision IMU and the auxiliary agents must communicate with the anchor agents to localize themselves.

In this paper, we propose a Multi-Agent Deep Reinforcement Learning based architecture called the Graph Localized Proximal Policy Optimization (GALOPP) to perform persistent monitoring with such heterogeneous agents subject to localization and communication constraints. The dynamic environment is modelled as a two-dimensional discrete grid. Whenever a cell in the grid is not monitored by any agent, then a penalty is allocated to that grid. The penalty accumulates as the cell is neglected with time. If the cell is in the sensor range of any agent then the penalty is zero. Therefore, the agents must learn their motion strategy such that the penalty accumulated is minimized. When an auxiliary agent gets disconnected from an anchor agent, they continue to update the positional beliefs using Kalman filters; however, the measurement sensors of such agents are deactivated and they do not contribute in updating the collective reward. This constraint is introduced to avoid an unlocalized agent updating uncertain information into the global estimates of the robotic team. This challenge, enforces the agents to stay localized to an anchor agent in order to obtain useful information. In an environment, where the joint actions of the agents influence the decision of others, the agents must also learn to effectively communicate their observations and their policies. To achieve this, the graph localization based architecture effectively facilitates checks for agent-to-agent connectivity as they perform surveillance of the environment.

The main contributions of the paper are (i) MADRL based architecture GALOPP that determines policies for the agents taking the localization and communication constraints and the robot heterogeneity into account (ii) a framework to utilize a combination of anchor and auxiliary agents to achieve persistent monitoring, and (iii) compare the performance of GALOPP to random and greedy baseline strategies. The results show that GALOPP outperform both these baseline strategies.

2 Related Work

In the literature, persistent monitoring and cooperative localization have been addressed as individual topics of interest and there are inadequate works that consider these two aspects jointly. The
mobile variant of the Art Gallery Problem (AGP) [12] details the movement of multiple agents in a
deterministic interior line segment. The goal is to determine the upper bound on the minimum num-
ber of agents required for patrolling the path/segments so as to minimize the time taken to cover the
entire area. Also, the Watchman Route Problem (WRP) [13] gives an algorithm to find the minimum
length trajectory for a watchman (robot) to cover every point inside the input polygon. Our work
is on developing a learning-based strategy to solve for multiple watchmen (robots) in the MADRL
framework. In addition, there have been several deterministic variants of multi-robot planning algo-
rithms within a dynamic environment [1, 2, 5, 14]. In general, the deterministic visibility coverage
problems are NP-hard and can provide only an approximation of the optimal solution. However,
these static policies do not work well for dynamic environments.

In MADRL setup, there have been numerous proposed studies on cooperative multi-agent tasks
covering a wide spectrum of applications in deep learning [15, 16, 17, 18, 19, 20]. In Chen et al. [7],
a method to find the trajectories for each agent to continuously cover the dynamic area is developed.
However the agents do not have communication and localization constraints.

For determining coordination strategies, it is essential for robots to incorporate the positional beliefs
of surrounding agents and obstacles. Previously, there have been several works describing methods
to achieve cooperative localization under limited inter-vehicle connectivity [9, 10, 21, 22]. Extensive
work has been done on achieving cooperative localization using Kalman filters [23]. In our study,
we incorporate the idea of cooperative localization using KF to achieve localization.

3 Problem Statement

3.1 Formal description of PM

We consider the PM problem for a 2D grid world $G \subseteq \mathbb{R}^2$ of size $M \times M$. Each grid cell $G_{\alpha \beta}$,
$1 \leq \alpha \leq M$ and $1 \leq \beta \leq M$, has a reward $R_{\alpha \beta}(t)$ associated with it at time $t$. When the cell $G_{\alpha \beta}$ is
within the sensing range of an agent, then $R_{\alpha \beta}(t) \leftarrow 0$, otherwise, the reward decays with a decay
parameter $\Delta_{\alpha \beta} > 0$ until it reaches a minimum value of $-R_{\text{max}}$. We consider negative reward as it
refers to penalty on the cell for not monitoring. At time $t=0$, $R_{\alpha \beta}(t)=0$, $\forall (\alpha, \beta)$ and $R_{\alpha \beta}(t+1) =
\max \{ R_{\alpha \beta}(t) - \Delta_{\alpha \beta}, -R_{\text{max}} \}$ if $G_{\alpha \beta}$ is not monitored at time $t$; else $R_{\alpha \beta}(t+1) = 0$ if $G_{\alpha \beta}$ was
monitored at time $t$ [7].

As the rewards are modelled as penalties, the objective of the PM problem is to find a policy that
would not neglect any cell for too long, thereby minimizing the net penalty accumulated by $G$, over
a finite time $T$. The optimal policy is given as

$$
\pi = \arg \max_{\pi} \sum_{t=0}^{T} \left[ \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} R_{\alpha \beta}(t) \right],
$$

where $R_{\alpha \beta}^\pi$ is the reward due to following a policy $\pi$.

3.2 Localization for Persistent Monitoring

The grid $G$ consists of $N$-agents to perform the monitoring task, with each agent being able to
observe a sub-grid of size $l \times l$, (for $l < M$), centred at its own position, and having a communication
range $\rho$. At every time step, a connectivity graph $G = (V, E)$ is generated between the agents. An
edge connection $e_{ij}$ is formed between agents $i$ and $j$ if they are separated by a distance within the
communication range, $\text{dist}(i, j) \leq \rho$. The connectivity of any agent with an anchor agent, is checked
by using Depth-First Search (DFS) algorithm. If an auxiliary agent is connected to an anchor agent,
either through a direct connection or via a k-hop connection, it can localize itself. This is possible
because (1) all the agents are aware of their true positions with 100% belief at the beginning of the
episode, and (2) the anchor agents have a high precision IMU with no error, so the anchor agents
are always localized, hence other agents can calculate their own positions by sensing their relative
position with respect to an anchor.

An agent observes the portion of the grid world that lies within its visibility range ($l \times l$), and
accordingly updates the rewards $R_{\alpha \beta}(t)$ in the grid world $G$, i.e. set $R_{\alpha \beta}(t) = 0$. We only allow
the anchor and the auxiliary agents that are connected to an anchor at time $t$, to reset the rewards
of the observed grid cells. If an auxiliary agent is unlocalized, then it cannot reset the rewards of the observed grid cells to 0. We take this constraint into account to address the problem where an unlocalized agent updates the rewards of the cells which are not actually within the observable range of its true position, but because of a low-precision IMU. The auxiliary agents use a Kalman filter (details of which have been provided in supplementary material) to estimate and update their positional belief when they are disconnected from an anchor and to rectify their state estimate when they reconnect to an anchor agent.

4 Graph Localized PPO - GALOPP

Given the model of the environment, the agent localization and communication constraints defined as above, we now develop the GALOPP framework for the agents to determine their policies.

4.1 Architectural overview

The GALOPP architecture as shown in Figure 2a consists of a multi-agent actor critic model that implements Proximal Policy Optimization (PPO) [24] to determine individual agent trajectories. The agent’s observation space is the shared global reward map passed to each agent. The decentralized actors of each agent take the generated embedding from the agents to learn the policy, while a centralized critic updates the overall value function of the environment. The model uses Convolutional Neural Network (CNN) [25] to generate the individual embeddings which is then augmented with agent $i$’s positional mean $\mu_i$ and covariance $\Sigma_i$. This serves as the complete information $z_i$ of the agent’s current state. The Graph Attention Network (GAT) [26] enforces the relay of messages in the generated connectivity graph $\mathcal{G}$ to ensure inter-agent communication. The model is trained end-to-end for the PM problem. The components of the GALOPP architecture is described in the below subsections.

4.2 Embedding extraction and message passing

The GALOPP model takes the shared global reward values in the 2D grid as input. The observation of an agent $i$ at time $t$ is the set of cells that are within the sensing range (termed as local map) and also a compressed image of the current grid (termed as mini map) with the pixel values equal to the penalties accumulated by the grid cells [7]. The mini map is resized to the shape of the local map of the agent and then concatenated to form a 2-channeled image (shown in figure 2b). This forms the input $o_i$ for the network at time $t$. The CNN is used to process and extract the embedding vector $h_i$ from input $o_i$. The position $p_i = [\mu_i, \Sigma_i]^T$ of each agent is concatenated to the generated embedding to form the new information vector $z_i = h_i||p_i$.

The GAT is used to transfer the information vector $z_i$ to all agents within the communication graph. Agents $i$ and $j$ are connected if the condition $\text{dist}(i, j) < \rho$ is satisfied. The agents take in the
weighted average of the embeddings of the neighbourhood agents. The attention parameter $\alpha_{i,j}$ gives an implicit weight parameter that assigns an attention value to each edge in the graph. The dynamics of the attention parameter $\alpha_{i,j}$ is given as [26],

$$\alpha_{i,j}^m = \frac{\exp(\text{LEAKYRELU}(a(W^m z_{i}, W^m z_{j})))}{\sum_{z_{j} \in \mathbb{R}^F} \exp(\text{LEAKYRELU}(a(W^m z_{i}, W^m z_{j})))}$$  \hspace{1cm} (2)

where, $z_{i} \in \mathbb{R}^F$ is the information vector of agent $i$; $N_{i}$ is the neighbourhood set of agent $i$; $W$ is the corresponding weight parameters for the inputs; $a: \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}$ is a single-layered feedforward function known as the attention mechanism; $m$ is the number of attention heads in the network used to stabilize the training process and to encompass complex parameter relations. LEAKYRELU is the activation function used on the output of $a$. After the message passing, the aggregated information vector $z'$ for each agent $i$ is given as,

$$z'_i = 1/M \sum_{m=1}^{M} \left( \sum_{k \in N_i} \alpha_{i,j}^m W^m z_k \right).$$  \hspace{1cm} (3)

**Algorithm 1**: Proximal Policy Optimization for multiple agents

**Input**: Initialized policy weights $\theta$, number of agents $K$, and number of episodes $E$

**Set Params**: Discounting factor $\gamma$, epochs $\kappa$, episode length $T$

for $E$ iterations do
  for $i=1,2,...,T$ do
    Collect the set of $\{s^i_t, a^i_t, r^i_t, s^i_{t+1} \}$ for $k \in [1, 2, ..., K]$ using policy $\pi_0$
  end
  Compute discounted reward at time $t$ as $\hat{R}_i = \sum_{\tau=t}^{T} \gamma^{T-\tau} R(\tau)$, for $\tau \in [1, ..., T]$
  for $\kappa$ iterations do
    Compute the advantage estimate $\hat{A}(s_t, a_t)$ and log probability $log \pi_\theta(a_t | s_t)$ for time $t$
    Compute clipped objective function $L(\theta)$
    Back propagate and update the policy weights $\theta \leftarrow \theta + \nabla_\theta \log \pi_\theta(a_t | s_t) \hat{A}(s_t)$
  end
end

4.3 Multi-agent actor critic method using PPO

The goal of Proximal Policy Optimization (PPO) is to incorporate the trust region policy [27] constraint in the objective function. Multi-agent PPO is preferred over other policy gradient methods to avoid having large policy updates and to achieve better stability in learning in monitoring tasks. The decentralized actors in the multi-agent PPO take in the aggregated information vector $z'_i$ and generate the corresponding action probability distribution. The centralized critic estimates the value function of the environment to influence the policy of the individual actors. The shared reward for all agents is defined in Equation (1).

The multi-agent PPO algorithm is defined in Algorithm 1. For a defined episode length $T$, the agent interacts with the environment to generate and collect the trajectory values in the form of states, actions, and rewards i.e. $\{s_t, a_t, R_t\}$. The stored values are then sampled iteratively to update the action probabilities and to fit the value function through back-propagation. The PPO gradient expression is the expected product of the advantage estimate function and the log probabilities.

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta (a_t | s_t) \hat{A}_i]$$  \hspace{1cm} (4)

where the advantage estimate function $\hat{A}_i$ is defined as the difference between the discounted sum of rewards ($Q(s_t, a_t)$) and the state value estimate ($V_i(s_t)$) [24].

$$\hat{A}_i(s_t, a_t) = Q(s_t, a_t) - V_i(s_t).$$  \hspace{1cm} (5)

The clipped surrogate objective function for a single PPO agent $i$ is given by (as defined in [24]),

$$L^C_{\text{CLIP}}(\theta) = \mathbb{E}_r \left[ \min \left( r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right].$$  \hspace{1cm} (6)
Figure 3: The 2-room and 4-room maps (along with their dimensions). The agents cannot move into black pixels, while the non-black regions needs to be persistently monitored. The anchor agents are depicted in red stars while the auxiliary agents are depicted in dark blue triangles. The fading white trails showing the trajectory followed by an agent in the last 30 steps. The red lines between 2 agents shows that they are within communicable range of each other.

where, $r_t(\theta)$ is the ratio of the action probability in the current policy to the action probability in the previous policy distribution for trainable parameter $\theta$. The $clip$ function clips the probability ratio $r_t(\theta)$ to the trust-region interval $[1 - \epsilon, 1 + \epsilon]$ [27]. The final modified multi-agent PPO objective function to be minimized in the GALOPP network is given as:

$$L(\theta) = \frac{1}{m} \sum_{m} \left( \frac{1}{N} \sum_{i=1}^{N} (L_{CLIP}^i(\theta)) \right)$$ \hspace{1cm} (7)$$

where, $N$ is the total number of agents and $m$ is the mini-batch size.

5 Experiments and analysis

To evaluate the performance of GALOPP, we custom-built two environments: the 2-room map and the 4-room map as shown in Figure 3a and 3b. The agents have a sensing range of 15 × 15 units. We use the accumulated penalty metric for the evaluation and also evaluate the effect of communication range on the performance. Key results are here while the supplementary material contains additional results in detail.

Training: For the 2-room case, the training is carried out for 30000 episodes, while its 50000 episodes for the 4-room map case with each episode length $T = 1000$ time steps. The penalties in the grid cells are updated with decay-rate of $\Delta_{alpha} = 1$, $\forall (\alpha, \beta)$. The maximum penalty a cell can have is $R_{max} = 400$. The total reward at time $t$ is defined as $R(t) = \sum_{\alpha, \beta} R_{\alpha\beta}(t)$. For every training episode, the agents are initialized randomly in the environment but localized.

The GALOPP architecture input at time $t$ is the image representing the state of the grid $G'$, resized to a 15 × 15 image using OpenCV’s INTER_AREA interpolation method, and concatenated with the local visibility map of the agent, forming a 15 × 15 2-channeled image. The action space includes 5 possible actions: front, back, left, right and stay. For testing the learned trajectories, we evaluate it for 100 episodes, each episode for $T = 1000$ time steps, in their respective environments. The reward for test episode $\tau$ is denoted by $\mathcal{R}_{\tau}$, and is the summation of the reward at each time step $t$ of the episode i.e. $\mathcal{R}_{\tau} = \sum_{t=1}^{T} \mathcal{R}(t)$ and the final reward $\mathcal{R}_{avg}$ after $n = 100$ episodes is calculated by averaging the rewards of each episode i.e. $\mathcal{R}_{avg} = \sum_{\tau=1}^{n} \mathcal{R}_{\tau}/n$. The $\mathcal{R}_{avg}$ is used to evaluate the performance of the model.

Evaluation: For the 2-room map, 3 agents (1 anchor and 2 auxiliary agents) with comm range 20 are deployed. For the 4-room map we deploy 4 agents (2 anchor and 2 auxiliary agents) with 20 units comm range. Figure 4a and 4b shows the training curves for both the environments.

Comparing the performance of GALOPP with non-RL baselines: We compare the performance of GALOPP with 3 non-RL baselines: Random Search (RS), Random Search with Ensured Communication (RSEC) and Greedy Decentralized Search (GDCS). In Random Search, each agent ran-
domly chooses from the available actions uniformly, without caring if an auxiliary agent gets unlocalized. In RSEC, every agent chooses a random action and checks if taking that action will lead to its unlocalization or the unlocalization of any other agent. If so, then it randomly chooses from the remaining actions and so on until it finds an action that won’t unlocalize itself or any other agent.

In GDCS, agents act independently and greedily. Given that an agent $i$ has an $l \times l$ visibility range, is currently located at position $(x, y)$ and $G'$ defines the set of grid cells that fall on the unobstructed line of sight of agent $i$, we define: $G = \{G_{\alpha\beta} \in G' | \alpha \in [x-(\frac{l}{2}+1),...,x+(\frac{l}{2}+1)], \beta \in [x-(\frac{l}{2}+1),...,x+(\frac{l}{2}+1)], (\alpha, \beta) \neq (x \pm (\frac{l}{2}+1), y \pm (\frac{l}{2}+1))\}$, which is a set of cells that are just one step beyond agent $i$’s visibility range. $A_i$ chooses an action that takes it towards the cell with the maximum penalty in $G$. If all the grid cells in $G$ have the same penalty, then $A_i$ chooses a random action. GDCS does not impose the communication constraint while taking an action.

Figure 5a and 5b compare the performance of our architecture with the baselines. In the 2-room map case, Figure 5a, GDCS and GALOPP always outperform the random baselines (RS and RSEC) by a significant margin. But coming to GDCS, its performance is close to that of GALOPP, with certain instances where it performs better than our algorithm (hence the high standard deviation on the GDCS bars in Figure 5a). Our tests revealed that the GDCS algorithm is highly susceptible to the initialization positions of the agents. GDCS only performs well when the agents are initialized in a manner where most of the cells fall within the unobstructed line of sight of the agents (this happens mainly when the agents are initialized near to or in the central corridor region of the 2-room map. But when the agents are initialized in an unfavourable location like in corners of one of the rooms, then GDCS leads to a situation where the agents are stuck leading to sub-optimal trajectory that propagates to some of the grid cells reaching $R_{max}$ in their penalties.

However, GALOPP can adapt to random initialization positions and plan the trajectories accordingly. In the 2-room map we notice that our algorithm ends up with the agents in a formation where two of the agents position themselves in the 2 rooms while one agent monitors the corridor. This can be seen in Figure 3a where the faded cells show the trajectory followed by the agents.

Figure 5b compares the performance of GALOPP in the 4-room map. We notice that in this case, the GDCS is outperformed by the random baselines and GALOPP. The poor performance of the GDCS algorithm can be attributed to the narrow passages that lead to the individual rooms. This obstructs the view of the agents into the grid cells within the room, hence leaving a subset of rooms unmonitored. This comes to show that GDCS does not adapt well to complex environment apart from being highly dependent on the agents’ initialization locations. GALOPP performs better than the random baselines as well, with a significant improvement in performance and the ability to adapt to complex environments. We see that even in this case, our algorithm learns a trajectory to maintain a formation where each of the 4 agents, monitor a room and they intermittently exit the room to monitor the central corridor region (shown in Figure 3b).

**Effect of varying communication range:** We analyze the effect of changing the communication range $\rho$ to the overall performance. Initially, for the 2-room map communication ranges from 15 to 25 units (pixels) are considered, while in the 4-room map $\rho$ from 20 to 40 units are considered.
Figure 5: Test results (units $10^8$) for comparing the performance of GALOPP with non-RL baseline, over varying communication ranges, in (a) 2-room map and (b) 4-room map.

In the 2-room map (Figure 5a) we notice that the model with $\rho = 15$ performs worse while the models with $\rho = 20$ and $\rho = 25$ are almost equal in their performance. On the other hand, for the 4-room map (Figure 5b) we notice consistently similar performance for all the communication ranges.

The figure also shows that beyond a certain minimum $\rho$, GALOPP performs almost identically similar with increasing communication range. This is also intuitive because as $\rho$ increases, it will only facilitate the localization of the auxiliary agents at further distances from the anchor agents where as the optimum policy would not require it to go to such large distances.

Figure 6 shows the average percentage of time an agent is unlocalized during an episode. This is calculated by $U = \sum_{i=1}^{K} U_i / K$ where $K$ is the total number auxiliary agents and $U_i$ the total unlocalization time for each auxiliary agent in an episode. This is then repeated for all the 100 episodes and averaged over them. In case of the 2-room map we see high unlocalization times for $\rho = 15$ followed by a drop for $\rho = 20$ and then a rise again when $\rho = 25$. Even though the unlocalization times for $\rho = 25$ are high, Figure 5a shows that it performs similar to the model with $\rho = 20$. This suggests the emergence of policies that might not require localization of the auxiliary agents during the entire episode while still performing optimally. For the 4-room map on the other hand (Figure 5b), there is a steady drop in the unlocalization time as the comm range increases and its unlocalization time is far less when compared to the same comm range in the 2-room map. This is because the 4-room map had 2 anchor agents which lead to higher localization time for the auxiliary agents, as opposed to the 2-room map that only had 1 anchor agent.

6 Conclusion

This paper developed a MADRL based algorithm – GALOPP for persistently monitoring a dynamic region taking the communication, sensing and localization constraints into account. The agents learnt the dynamic environment and modified their trajectories to satisfy the conflicting objectives of coverage and meeting localization constraints based on the dynamic environmental conditions. The experiments show that the agents using GALOPP outperformed the greedy and random baseline strategies. This work can be further work extended to study its scalability to larger number of agents, robustness to varying dynamic environments and resilience in case of agent failures.
References


