Multi-Agent Area Coverage Control using Reinforcement Learning

Presented by Simon Hu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Problem Statement

Consider a group of N homogeneous agents moving in a compact environment $\Omega \subset \mathbb{R}^2$ where the dynamics of the agent are given by $\dot{p}_i = g(p_i, u_i)$ where $p_i = (x_i, y_i) \in \mathbb{R}^2$ is the agent position and $u_i = (u_{x_i}, u_{y_i})$ represents the control input. The goal is to find a configuration of agent positions $p = (p_1, p_2, \ldots, p_N)$ such that the cost index

$$\mathcal{H}(p,t) = \int_\Omega \max_{i=1,\ldots,N} f_i(\|p_i - q\|)\phi(q,t)\,\mathrm{d}q$$

is maximized. Though classical control methods have been proposed for solving the problem, reinforcement learning methods are more robust. Applications include search and rescue, data collection, and surveillance missions.

Voronoi Partitions

The Voronoi partition of Ω is given by $\mathcal{V} = \bigcup \mathcal{V}_i$ where each $\mathcal{V}_i$ is given by

$$\left\{ q \in \Omega \mid \|q - p_i\| \le \|q - p_j\|, \forall j \neq i \right\}.$$

The mass $m_{\mathcal{V}_i}$ and center of mass $c_{\mathcal{V}_i}$ of each V_i are given by

$$m_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \phi\,\mathrm{d}q, \quad c_{\mathcal{V}_i} = \frac{1}{m_{\mathcal{V}_i}} \int_{\mathcal{V}_i} q\phi\,\mathrm{d}q$$

Then the cost index $\mathcal{H}$ can be rewritten as

$$\mathcal{H}(p,t) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} f_i(\|p_i - q\|)\phi(q,t)\,\mathrm{d}q.$$

[Cortés et al., 2004] showed that the optimal partition of Ω that maximizes $\mathcal{H}$ is the centroidal voronoi partition.

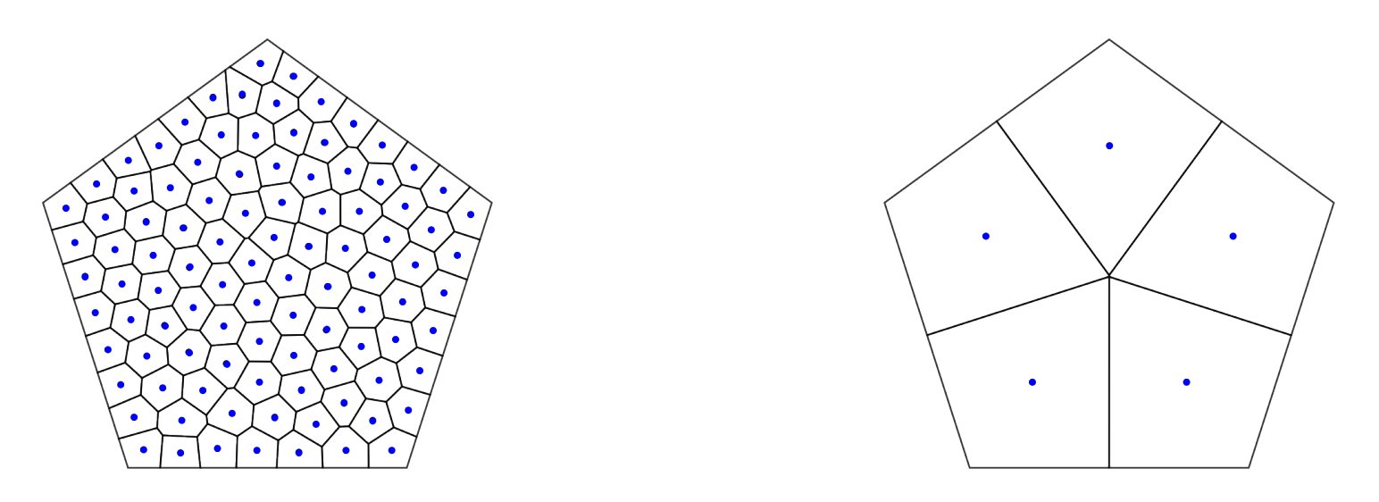

Figure 1: Example of a Voronoi partition, and a centroidal Voronoi partition, for $\phi(q,t) = 1$.

TD3 Algorithm

Algorithm 1 Twin-Delayed Actor-Critic DDPG

```
1:  Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network $\pi_\phi$ with random parameters $\theta_1, \theta_2, \phi$.
2:  Initialize the target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$.
3:  Initialize a replay buffer $R$.
4:  for $t = 1$ to $T$ do
5:      Select action with exploration noise $a \sim \pi(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$ and record the reward $r$ and new state $s'$.
6:      Store the tuple $(s, a, r, s')$ into $R$.
7:      Sample minibatches of $N$ transitions $(s, a, r, s')$ from $R$.
8:      Smooth the target policy according to $\tilde{a} \leftarrow \pi_{\phi'}(s) + \epsilon, \epsilon \sim \mathrm{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$.
9:      Perform double $Q$-learning and clip the results according to the followng rule $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$.
10:     Update the critics according to $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$.
11:     if $t \mod d$ then
12:         Update $\phi$ by the deterministic policy gradient according to the following rule $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\big|_{a=\pi_{\phi(s)}} \nabla_\phi \pi_{\phi(s)}$.
13:         Update the target networks according to $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i', \phi' \leftarrow \tau\phi + (1-\tau)\phi'$.
14:     end if
15: end for
```

The value function is given by

$$V(e_i(k)) = \sum_{\kappa=k}^{\infty} e_i^T(\kappa) Q e_i(\kappa) + u_i^T(\kappa) R u_i(\kappa)$$

The value function and policy $\pi_i = u_i$ are approximated by a neural network with weights $\omega_{c,j}$ and $\omega_{a,j}$, and activation functions ρ and σ:

$$\widehat{V}_j(e_i(k)) = \omega_{c,j}^T \rho(e_i(k)), \;\; \widehat{u}_j(e_i(k)) = \omega_{a,j}^T \sigma(e_i(k)).$$

The loss for the critic and actor networks are given by the MSE loss.

Experimental Setup

The dyanmics are given by $\dot{p}_i = u_i$ with maximum velocity 1 m/s. The learning rate used is $3 \times 10^{-4}, 3 \times 10^{-3}$ for the actor and critic respectively. Both networks have two hidden layers of size 400 and 300. ReLU activation is used at each layer and a tanh activation is used at the output for the actor network only. A discount factor of $\gamma = 0.99$ is used and the soft update parameter $\tau = 0.001$ is used. A predictor-corrector method is used for numerical integration.

20 Agents are deployed in a pentagonal environment, described by $(0, 100), (-95, 31), (-59, -81), (59, -81), (95, 31)$, and three scenarios are considered. In all three scenarios, different combinations of Gaussians place at different locations with varying covariances. The covariances in the x and y directions are assumed to be the same in scenarios 1 and 2, but not in scenario 3. Additionally, the Agents are initialized by uniformly sampling points within a rectangular area near the bottom of the pentagon.

Results

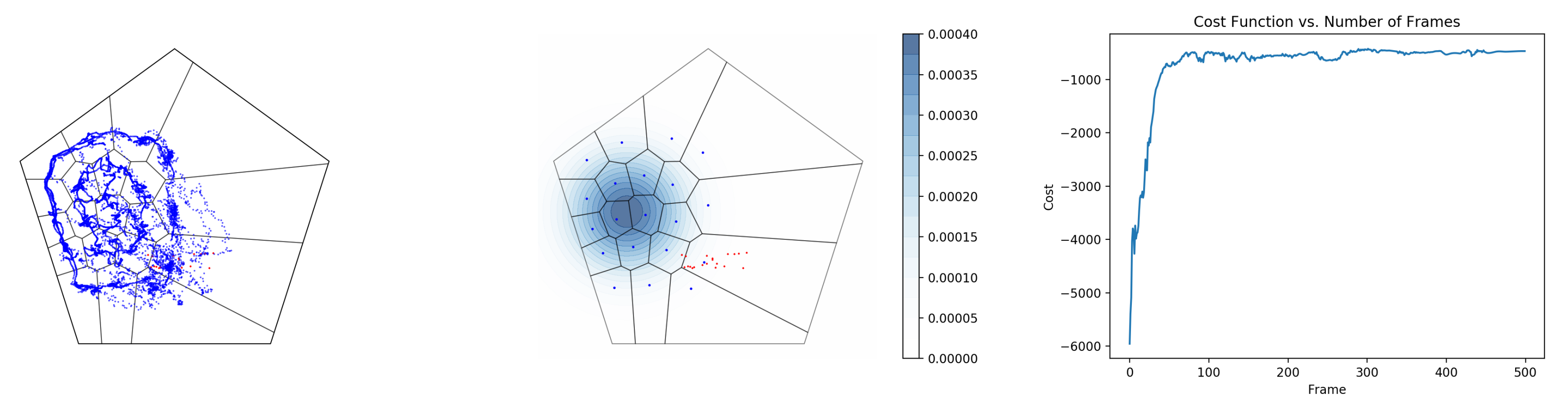

Figure 2: Scenario 1: The position of the agents is tracked (left), with the red dot denoting the starting point. The final configuration of the agents is also shown (center). The cost function $\mathcal{H}$ as a function of the number of frames.

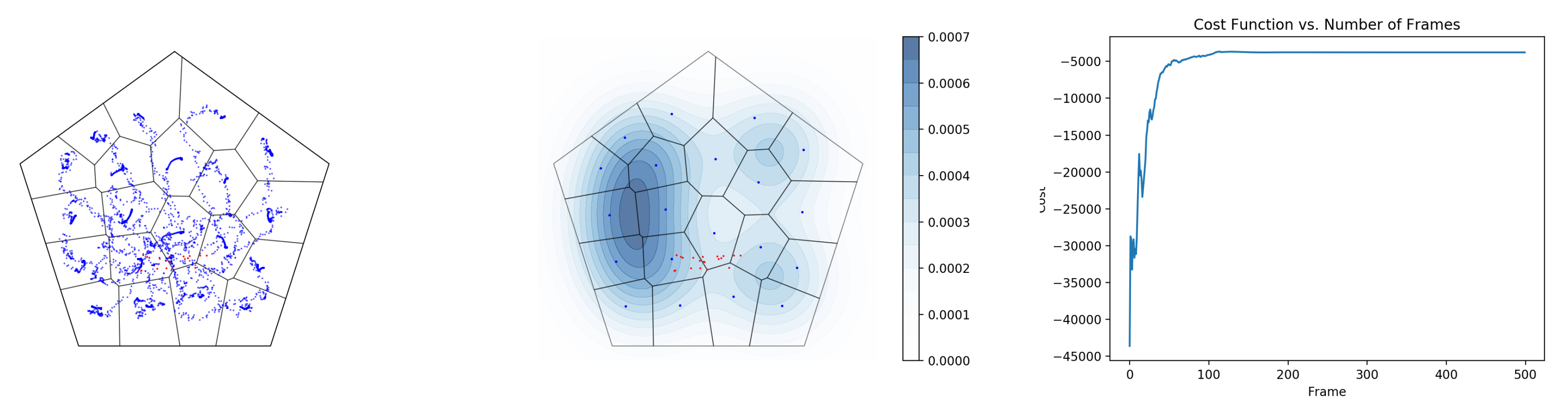

Figure 3: Scenario 2. There is good coverage at the dark blue areas, which is the expected result.

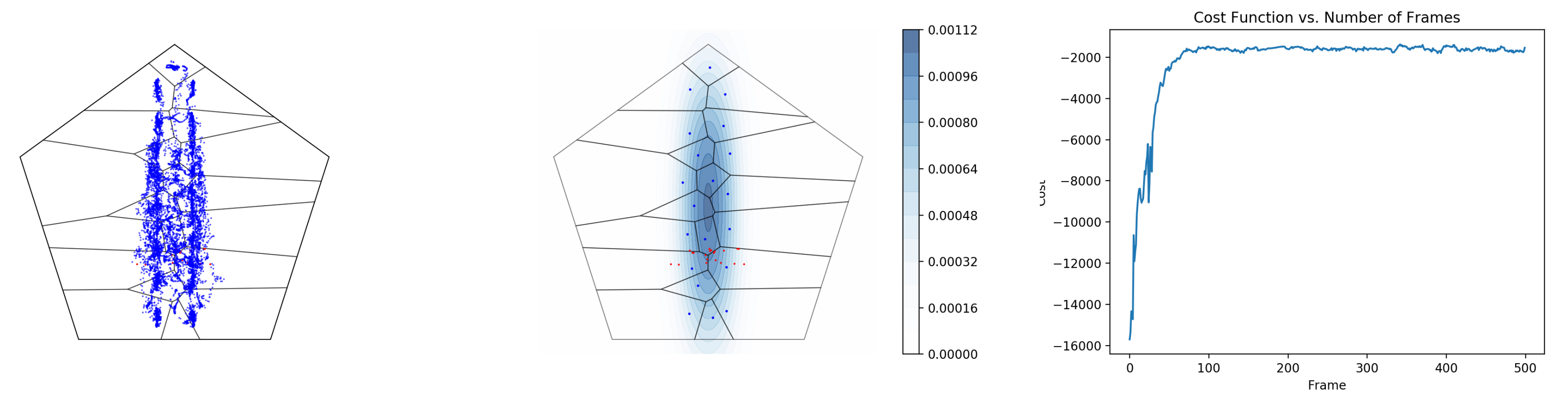

Figure 4: Scenario 3. There is good coverage on the center strip, which is the expected result. However, the cost function is high relative to the number of agents and area to cover.

References

References

[1] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, April 2004.

[2] A. Adepegba, S. Miah, and D. Spinello, "Multi-agent area coverage control using reinforcement learning," 2016.

[3] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," pp. 1039–1044, June 2011.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.