Summary of Multi-Agent Area Coverage Control Using Reinforcement Learning

Simon Hu

1 Introduction

Coverage control of mutli-agent systems is concerned with deploying agents over an environment to maximize sensor coverage, for various tasks such as sensing, data collection and surveillance. Consider a group of N homogeneous agents moving in a compact environment $\Omega \subset \mathbb{R}^2$, where the dynamics of the i-th agent is given by

$$\ddot{p}_i = u_i \tag{1}$$

where $p_i = (x_i, y_i) \in \mathbb{R}^2$ represents the agent's location and $u_i = (u_{x_i}, u_{y_i}) \in \mathbb{R}^2$ represents the velocity vector. The goal of solving the coverage control problem is to find a configuration of agent positions, $p = (p_1, p_2, \ldots, p_N)$ such that the cost index

$$\mathcal{H}(p,t) = \int_\Omega \max_{i=1,2,\ldots,N} f_i(\|p_i - q\|)\phi(q,t)\ \mathrm{d}q \tag{2}$$

is maximized. In the above, $\phi : \Omega \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a probability density that represents the probability of an event occuring at point $q \in \mathbb{R}^2$ and $f : \mathbb{R}^2 \to \mathbb{R}$ is a non-increasing Lesbegue measurable function of the distance between points. Intuitively, ϕ reflects some measure of the number of agents that need to be deployed to a particular point. The choice of f reflects the sort of coverage task we are attempting to achieve. For maximum area coverage, the function $f(x) = -x^2$ is considered.

A collection $S = (S_1, S_2, \ldots, S_M)$ is a partition of Ω if $\mathrm{int}(S_i) \cap \mathrm{int}(S_j) = \emptyset$ (i.e. the sets have disjoint interiors) and $\cup(S_i) = \Omega$ (i.e. the union of the sets is Ω). The Voronoi partition of Ω is given by $\mathcal{V}s = (\mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^N)$ where each $\mathcal{V}_i$ is described by

$$\mathcal{V}_i = \left\{ q \in \Omega \mid \|q - p_i\| \leq \left\|q - p_j\right\|, \forall j \neq i \right\}. \tag{3}$$

Under this partition, the cost index (2) can be rewritten as

$$\mathcal{H}(p,t) = \sum_{i=1}^{n} \int_{\mathcal{V}_i} f(\|q - p_i\|)\phi(q,t)\ \mathrm{d}q. \tag{4}$$

In this manner, the algorithm is spatially distributed, i.e. each agent only has to use local neighbor information to compute its contribution to the total cost. The mass and center of mass of $\mathcal{V}_i$ are given by

$$m_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \phi(q)\ \mathrm{d}q, \quad c_{\mathcal{V}_i} = \frac{1}{m_{\mathcal{V}_i}} \int_{\mathcal{V}_i} q\phi(q)\ \mathrm{d}q \tag{5}$$

respectively. It is shown in [1] that the optimal partition of Ω is the Voronoi partition and the optimal sensor placements, p are the centers of mass of the Voronoi cells.

2 Actor-Critic Neural Network

Designing a controller that realizes the control law considered in [2] is difficult, so an Actor-Critic Neural Network (ACNN) approximation is employed. Define $e_i(t) = c_{\mathcal{V}_i}(t) - p_i(t)$ as the centroid error for agent i and consider the Value function at the k-th step, given by

$$V(e_i(k)) = \sum_{\kappa=k}^{\infty} e_i^T(\kappa) Q e_i(\kappa) + u_i^T(\kappa) R u_i(\kappa) \tag{6}$$

where $Q, R \in \mathbb{R}^{2\times 2}$ are positive definite matrices. It is important to note that the second term in the Value function quantizes some measure of energy expenditure, which is important for long-term surveillance and data collection tasks. For notational simplicity, let us write $V_{i,k} = V(e_i(k))$. To put the Value function into the form of a Bellman equation, we rewrite (6) as

$$V_{i,k} = e_i^T(k) Q e_i(k) + u_i^T(k) R u_i(k) + V_{i,k+1} \tag{7}$$

so that the minimization to be solved is

$$V_{i,k}^* = \min_{u_i(k)} \left[e_i^T(k) Q e_i(k) + u_i^T(k) R u_i(k) + \gamma V_{i,k+1}^* \right] \tag{8}$$

where γ is the discount factor. To put this in the language used in the class, we are trying to find the optimal policy u_i that minimizes the value function. To solve this problem, we assume the following structure for the Value function and policy.

$$\widehat{V}_j(e_i(k)) = w_{c,j}^T \rho(e_i(k)), \quad \widehat{u}_j(e_i(k)) = w_{a,j}^T \sigma(e_i(k)). \tag{9}$$

Here, $\widehat{V}_i$ and $\widehat{u}_i$ are the estimates of the Value function and policy respectively, $w_{c,j}, w_{a,j}$ are the network weights, and ρ, σ are the activation functions of the

network. The loss of the critic and actor neural networks are given by

$$E_{c,j} = \frac{1}{2}\sum_{\ell=1}^{N_t}\left[w_{c,j+1}^T \rho^\ell(e_i(k)) - V_{i,j+1}^\ell\right]^2, \;\; E_{a,j} = \frac{1}{2}\sum_{\ell=1}^{N_t}\left[w_{a,j+1}^T \sigma^\ell(e_i(k)) - u_{j+1}^\ell(e_i(k))\right]^2 \tag{10}$$

which is again just the least squares loss. A target and source network is initialized and tracked for both the actor and critic. The weights are synced (i.e. there is a soft update of the weights) to improve numerical stability by using a soft actor-critic update law.

3 Algorithm

With very minor modifications, the main meat of the algorithm is the Soft Actor-Critic DDPG algorithm. For more information, consult [3].

4 Simulation Results

References

[1] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, vol. 48, pp. 1077–1087, June 2012.

[2] 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.

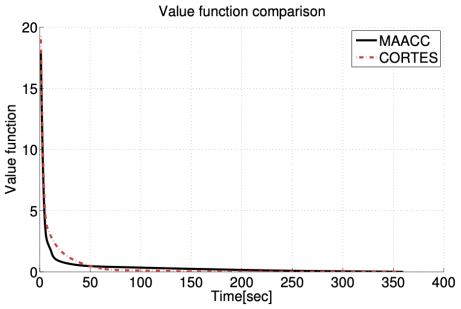
[3] "Deep deterministic policy gradient."

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

(a) The final, optimal deployment of five agents in a convex environment with the corresponding Voronoi cells and risk density function ϕ, in purple.

(b) The percentage of coverage as a function of time. The black line shows the results from [4] (i.e. this paper) and the dash red line shows the results from [2].

(c) The Euclidean error between each agent and its corresponding centroid.



(d) The output of the value function. The black line shows the results from [4] (i.e. this paper) and the dashed red line shows the results from [2].