Risk-Averse Sensor Planning using Distributed Policy Gradient

Wann-Jiun Ma, *Member, IEEE*, Darinka Dentcheva, Michael M. Zavlanos, *Member, IEEE*

***Abstract*— This paper considers a risk-averse approach to planning the motion of mobile sensor networks in order to maximize the information they collect in uncertain environments. Recent models of risk shape the tails of the probability distributions of the decision variables, controlling in this way the occurrence of rare but important events. In this paper, we formulate the sensor planning problem as a Markov Decision Process (MDP) and propose a distributed risk-averse policy gradient method to obtain optimal policies for the team of sensors. These policies avoid extremely low reward and high risk events. The simulation results validate the effectiveness of the proposed distributed risk-averse method.**

I. Introduction

In this paper, we consider mobile sensor networks operating in uncertain environments, that are tasked with collaboratively maximizing information acquisition while avoiding dangerous regions, e.g., regions that contain obstacles or where the sensors can be detected by adversary agents. Such planning problems are often formulated in the literature using Markov Decision Processes (MDPs) [1], where the state space contains the possible configurations of the sensors, the actions allow the sensors to move between adjacent configurations in the state space, and the reward function rewards information acquisition and penalizes actions that drive the sensors to dangerous regions. The goal is to obtain an optimal policy that allows the sensors to accomplish their assigned tasks and maximize the total reward. Standard dynamic programming methods, such as policy or value iteration, can be used for this purpose [1]. Nevertheless, if the state space of the MDP is too large or unknown, approximation methods are necessary to address computational complexity [5]. For example, this is the case if continuous variables are incorporated in the state space.

Reinforcement learning [4, 9]–[11] is a typical approach to determine optimal policies for MDPs with models that are not known but instead learned along the way. Quite often, reinforcement learning problems are solved using policy gradient algorithms [9]. In these algorithms, MDP policies are appropriately parameterized and the gradient of the objective function of the MDP with respect to that parameter is estimated. Using the estimated gradient, the algorithm updates the parameter until a locally optimal solution is obtained consisting of the optimal policy and the associated objective value of the MDP. In this paper, we develop a distributed policy gradient method to guide the team of sensors in the environment in order to optimally acquire information. A similar approach is proposed in [5] where the authors consider a team of robots tasked with exploring the workspace and learning the MDP model collaboratively. Each robot can communicate with its neighboring robots, and the learning updates carried out by the robots are coordinated using a consensus procedure [13]. By exploring the workspace in parallel and allowing the team of robots communicate with each other, the robots can more efficiently learn the desired process.

A popular way to capture uncertainty in MDPs, that is common in the literature discussed above, is using the expectation. However, such mean-value formulations are typically unable to capture high risk events that have low frequency of occurrence but large impact on the system. Instead, here we consider models of risk [6]–[8] which aim at capturing the impact of low-probability but low-reward events, and develop a risk-averse, distributed policy gradient method to guide the team of sensors in an environment of interest in order to optimally acquire information. This work is most closely related to [5] and [12]. In particular, we incorporate in the risk-averse policy gradient algorithm proposed in [12] with the distributed actor-critic method in [5] and study convergence of the resulting method. To the best of our knowledge, this is the first distributed risk-averse reinforcement learning method for mobile sensor networks.

II. Problem Formulation

Consider a team of N mobile robot sensors tasked with monitoring and collecting information in an environment of interest. Let $\mathcal{N} := \{1, ..., N\}$ denote the set of robots. The workspace in which the sensors operate is described by a two-dimensional grid world. Grid points in the grid world represent possible positions of the sensors. Edges connecting adjacent grid points indicate motion steps that the sensors can take. While more general grid worlds are possible, in this paper we connect each grid point to its East, West, South, and North adjacent points. This means that at each time step, each sensor is allowed to move to the grid points that are East, West, South, or North of its current location.

At time $k \in \mathcal{K}$, the position of robot $i \in \mathcal{N}$ is denoted by $x_k^i \in \mathbb{R}^2$, where $\mathcal{K} = \{0, \ldots, K\}$ denotes the time horizon during which the sensors move. The actions available to the mobile sensors at this time instant are $u_k^i \in \mathcal{U} := \{(1,0), (-1,0), (0,0), (0,1), (0,-1)\}$, and correspond to steps in the East, West, North, and South directions, along

W.-J. Ma and M. M. Zavlanos are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708, USA, {wann.jiun.ma, michael.zavlanos}@duke.edu D. Dentcheva is with the Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ 07030, USA, darinka.dentcheva@stevens.edu. This work is supported in part by the NSF award CNS # 1302284.

with a "Stay" that keeps the robot at its current location. As a result of actions in $\mathcal{U}$, the position of the robot is updated as $x^i_{k+1} = x^i_k + u^i_k$, for $i \in \mathcal{N}$ and $k \in \mathcal{K}$.

In addition to the robots, we assume that there are M sources in the workspace that the sensors need to detect. Let $\mathcal{M} := \{1, \dots, M\}$ denote the set of sources. A source may be mobile and each source's position at time $k \in \mathcal{K}$ is denoted by m^i_k for $i \in \mathcal{M}$. Each robot has the ability to sense the intensity or signal strength of each source that is in its range. At time $k \in \mathcal{K}$, for source $j \in \mathcal{M}$, the strength of the signal received at position y is defined as

$$s^j_k(y) = \frac{w^j_k}{2\pi \det(\Sigma^j)^{1/2}} e^{-(y-m^j_k)^T(\Sigma^j)^{-1}(y-m^j_k)/2}, \quad (1)$$

i.e., it is assumed to be Gaussian with mean m^j_k and covariance Σ^j with a scale factor w^j_k. The signal strength of each source sensed by a robot can be used to design a motion strategy for that robot driving it to configurations that contain more information about the sources. In Section III, we address this planning problem for a single robot, while in Section IV, we develop a distributed solution where multiple sensors explore the workspace and estimate the sources in parallel, significantly increasing efficiency of the method.

III. Risk-Averse Markov Decision Process

We use Markov Decision Processes (MDPs) [1] to model the sensing task described in Section II. In our MDP, the state space $\mathcal{S}$ contains the robot coordinates in the 2-D grid world as well as the associated source signal strengths sensed by the robot at that location[1]. In particular, consider a state $q = (x, s) \in \mathcal{S}$, where $x \in \mathbb{R}^2$ represents the robot's position in the grid world and $s \in \mathbb{R}^M$ the signal strength received by the robot when it is located at position x, where M is the total number of the sources. The action space $\mathcal{U}$ is defined in Section II. Further, let $\mathbb{P}(u) := \Pr\{\cdot \mid \cdot,\ u\}$ denote the transition kernel depending on action u, for all $u \in \mathcal{U}$, which we assume that is not changing in time. The probability of transition from state $q \in \mathcal{S}$ to state $q' \in \mathcal{S}$ given action $u \in \mathcal{U}$ can be determined by the signal strength, robot location, and actions available to the mobile sensor. For example, assuming that the sources are static, one can use the signal received at each grid point and the location of the grid point to construct the probability of transition.

If the robot identifies source $j \in \mathcal{M}$, i.e., if $x_k = m^j_k$ at some time period k, then a random reward v^j_k is collected, whose distribution does not change in time; otherwise the reward is zero. We assume that different sources result in different rewards, modeling, e.g., different source intensities. Moreover, the positions of the sources m^j_k are assumed to be time-varying and random for all $j \in \mathcal{M}$. This allows us to model situations where uncertainty in the sensor measurements and outcomes is also present. Note that we assume that the reward can also take negative values to penalize for not detecting the source in time.

[1]In this section we focus on the motion planning of a single robot, therefore, for ease of presentation, we drop the superscript index i.

We denote by $Z := \sum_{k=0}^{K} R_k$ the total reward after K time steps, and denote by x^* a terminal state of the MDP. Here, we use upper case letters to denote random variables and lower case letters to denote realizations of the random variables, namely, R_k is the random reward collected by the robot at time $k \in \mathcal{K}$ and r_k represents a particular realization of R_k. Let $f_Z(z)$ and $\Psi_Z(z) = \Pr(Z \leq z)$ denote the probability density function and the cumulative distribution function (CDF) of Z, respectively. Assume that $\Psi_Z(z)$ is a continuous function of z. Similarly, denote by Ξ a random trajectory and the associated total reward of the MDP and by $\xi = (x_0, u_0, r_0, ..., x_K, u_K, r_K)$ one realization of the trajectory and reward Ξ, where the trajectory of a MDP is defined as a sequence of states and actions $(x_0, u_0, \cdots, x_K, u_K)$ of the MDP. Let $f_\Xi(\xi)$ denote the probability density function of Ξ.

A. Policy Gradient Approach

The first step in designing a policy gradient algorithm for our problem is to construct the parameterized MDP policy. Following [5, 12], we parameterize the policy $\mu_\theta(u \mid x_k)$ by a parameter $\theta \in \mathbb{R}^M$ for $k \in \mathcal{K}$ as

$$\mu_\theta(u \mid x_k) = \frac{\exp(\eta_\theta(u, x_k))}{\sum_{v \in \mathcal{U}} \exp(\eta_\theta(v, x_k))}, \quad (2)$$

where

$$\eta_\theta(u, x) = \sum_{j=1}^{M} \theta_j s^j_k(x + u). \quad (3)$$

The parameter θ is an M-dimensional vector and M is the total number of the sources. Each element θ_j of θ represents the weight applied to the signal s_j that is transmitted from source j, where $j \in \mathcal{M}$. The random policy $\mu_\theta(u \mid x_k)$ yields a probability distribution over possible actions given the current robot position x_k. Equation (2) represents a so called softmax policy [9]. The policy favors sources located close to the sensor and will drive the sensor towards those sources. In (3), $\eta_\theta(u, x)$ is a linear combination of the signal strengths s^j_k weighted by the weight vector θ.

Using the parameter θ, we can also parameterize the random vector Ξ. In particular, denote by $f_\Xi(\xi; \theta)$ the parameterized probability density function of Ξ. Similarly, we parameterize the random variable Z (the objective of the MDP) using the parameter θ. Let $f_Z(z; \theta)$ denote the parameterized probability density function of Z and $\Psi_Z(z; \theta)$ the cumulative distribution function of Z with a threshold $z \in \mathbb{R}$. We use the notation $\mathbb{E}[Z; \theta]$ to denote the expectation of Z given the parameter θ.

Given the parametrized MDP discussed above, we then take a sample of W trajectories $(\xi_1, ..., \xi_W)$ of the MDP and estimate the gradient of the expected objective $\mathbb{E}[Z; \theta]$ with respect to θ, where each sample $\xi \in \{\xi_1, ..., \xi_W\}$ consists of a state trajectory and the corresponding rewards, i.e., $\xi = (x_0, u_0, r_0, ..., x_K, u_K, r_K)$. Let $\bar{\Delta}_W := (\bar{\Delta}_{1;W}, ..., \bar{\Delta}_{M;W})$ denote the estimate of the gradient $\nabla \mathbb{E}[Z; \theta]$ with respect to the parameter θ, where we use subscript j in $\bar{\Delta}_{j,W}$ to indicate the j-th element of the vector $\bar{\Delta}_W$. Using the gradient estimate $\bar{\Delta}_W$ we can update the parameter θ in the gradient ascent direction in order to maximize the parametrized expectation of the cumulative rewards $\mathbb{E}[Z;\theta]$ as

$$\theta_{l+1} = \theta_l + \epsilon_l \bar{\Delta}_W, \tag{4}$$

where ϵ_l is a step size. Note that the subscript l in (4) indicates the iteration index of the gradient ascent algorithm. Each iterate θ_l is an M-dimensional vector and $\theta_{j,l}$ denotes the j-th element of θ_l. Under mild assumptions, the update (4) will lead the parameter θ to a locally optimal point [9].

In this policy gradient algorithm, the objective is defined as the expectation of the total rewards. The expectation may perform poorly due to the significant variability in the realizations of the random problem parameters. In the following, we replace the expectation operator with a risk model to improve the performance of the policy in highly uncertain environments with high variability.

B. Risk-Averse Policy Gradient

Compared to the expectation, risk models provide a solution, which is less sensitive to changes in the distribution of the decision variables without being overly conservative as in worst-case approaches that control the variance of the random variables. Also, unlike the expectation, also called a risk-neutral model, the risk models take into account the extreme low-reward events by shaping the tail of the probability distribution of the random decisions. There are various risk models proposed in the literature that have different properties and applications. Here, we use Conditional Value-at-Risk (CVaR) [2, 3, 6] which can be reformulated to obtain a convex form. Convexity of CVaR makes methods to control it computationally favorable compared to other risk models. Also, CVaR is a coherent risk measure with some nice properties. CVaR can be derived from the Value-at Risk (VaR) risk model. However, in general, VaR does not have a convex structure. Also, by tuning a parameter, CVaR can represent a broad spectrum of risk preferences (from the worst-case risk-averse to the risk-neutral preferences).

Specifically, the β-VaR of random variable Z (denoted by $\alpha_\beta(Z)$) is defined as

$$\alpha_\beta(Z) := \min\{z \in \mathbb{R} \mid \Psi_Z(z) \geq \beta\}. \tag{5}$$

The β-VaR represents the minimum value z such that for the corresponding CDF, it holds $\Psi_Z(z) \geq \beta$. Typical values of β are 0.05 or 0.1 [6]. The β-CVaR of random variable Z (denoted by $\phi_\beta(Z)$) is defined as the average of the values which are smaller than the β-VaR, i.e.,

$$\phi_\beta(Z) := \mathbb{E}[Z \mid Z \leq \alpha_\beta(Z)]. \tag{6}$$

Similarly, for the parameterized model discussed in Section III-A, let $\alpha_\beta(Z;\theta)$ and $\phi_\beta(Z;\theta)$ denote the parameterized VaR and CVaR, respectively.

By optimizing the CVaR $\phi_\beta(Z;\theta)$, one can efficiently shape the left tail of the probability distribution of Z given the decision made [6]. A risky event results in extreme low reward, and thus by appropriately shaping the left tail of the distribution of Z, events with the extreme low reward can be prevented in a risk-averse manner.

In what follows, we make some standard assumptions on the reward Z and the trajectory Ξ; see also [12].

Assumption 1: The total reward Z is a continuous variable, and bounded in $[-b, b]$ for all θ.

Assumption 2: For all θ and $1 \leq j \leq M$, the gradient $\frac{\partial \alpha_\beta(Z;\theta)}{\partial \theta_j}$ and $\frac{\partial \phi_\beta(Z;\theta)}{\partial \theta_j}$ are well defined and bounded.

Assumption 3: For all θ and $1 \leq j \leq M$, the gradient $\frac{\partial f_Z(z;\theta)}{\partial \theta_j} / f_Z(z;\theta)$ is well defined and bounded.

Assumption 4: $\frac{\partial \log f_\Xi(\xi;\theta)}{\partial \theta_j}$ is well defined and bounded for all ξ and θ.

If the reward contains only discrete values, one may add an arbitrarily small smooth noise to the total reward to ensure that Assumption 1 holds [12]. Also since Z is continuous, Assumption 2 is satisfied whenever $\frac{\partial f_Z(z;\theta)}{\partial \theta_j}$ is bounded [12]. Finally, Assumptions 3 and 4 are standard assumptions in reinforcement learning [12].

As shown in [12], the gradient of the CVaR can be expressed as a conditional expectation.

Proposition 1: [12] If Assumptions (1)-(4) hold, then

$$\begin{aligned}\frac{\partial \phi_\beta(Z(\Xi);\theta)}{\theta_j} \\ = \mathbb{E}\Bigg[\frac{\partial \log f_\Xi(\Xi;\theta)}{\partial \theta_j} (Z(\Xi) - \alpha_\beta(Z(\Xi);\theta)) \\ \Bigg| \, Z(\Xi) \leq \alpha_\beta(Z(\Xi);\theta) \Bigg].\end{aligned} \tag{7}$$

The result in Proposition 1 suggests a MC algorithm to estimate the gradient $\frac{\partial \phi_\beta(Z(\Xi);\theta)}{\theta_j}$ for $j = 1, ..., M$. In particular, we generate a sample of W trajectories $\xi_1, ..., \xi_W$ from $f_\Xi(\Xi;\theta)$ and estimate $\alpha_\beta(Z(\Xi);\theta)$, where each element in $\{\xi_1, ..., \xi_W\}$ represents one trajectory. As suggested in [12], $\alpha_\beta(Z(\Xi);\theta)$ can be estimated by the empirical β-quantile $\tilde{v}$

$$\tilde{v} = \inf_z \hat{\Psi}_Z(z) \geq \beta, \tag{8}$$

where $\hat{\Psi}_Z(z)$ is the empirical CDF of Z. One way to calculate the empirical β-quantile $\tilde{v}$ is by first sorting the $z(\xi_i)$'s in ascending order, $(z_1^s, ..., z_W^s) = \text{sort}(z(\xi_1), ..., z(\xi_W))$, and then setting $\tilde{v}$ as $\tilde{v} = z^s_{\lceil \beta W \rceil}$, where we let $z(\xi)$ denote the objective function value obtained by ξ, and the operator sort$(z(\xi_1), ..., z(\xi_W))$ basically sorts the elements in $\{z(\xi_1), ..., z(\xi_W)\}$ in ascending order. The MC estimates of the partial derivatives (denoted by $\Delta_{j;W}$) are then given as

$$\Delta_{j;W} = \frac{1}{\beta W} \sum_{i=1}^{W} \frac{\partial \log f_\Xi(\xi_i;\theta)}{\partial \theta_j} (z(\xi_i) - \tilde{v}) \mathbf{1}_{z(\xi_i) \leq \tilde{v}}, \tag{9}$$

for $j = 1, ..., M$, where $\mathbf{1}_{z(\xi_i) \leq \tilde{v}} = 1$ if $z(\xi_i) \leq \tilde{v}$, and $\mathbf{1}_{z(\xi_i) \leq \tilde{v}} = 0$, otherwise[2].

Remark 1: By the Markov property of the state transitions, the gradient of the probability density function $f_\Xi(\xi;\theta)$

[2]We use Δ_W and $\bar{\Delta}_W$ to denote the gradient estimates of the CVaR and expectation of the objective function, respectively.

```
Algorithm 1
Given: The CVaR level $\beta$, a sample of $W$ trajectories $\xi_1, ..., \xi_W$ of the process, and the probability density function $f_\Xi(\xi;\theta)$.
Output: The gradient estimate $\Delta_W := (\Delta_{1;W}, ..., \Delta_{M;W})$.
(1) $(z_1^s, ..., z_W^s) \leftarrow \text{sort}(z(\xi_1), ..., z(\xi_W))$,
(2) $\tilde{v} \leftarrow z^s_{\lceil \beta W \rceil}$,
(3) for $j = 1, ..., M$,
        $\Delta_{j;W} \leftarrow \frac{1}{\beta W} \sum_{i=1}^{W} \frac{\partial \log f_\Xi(\xi_i;\theta)}{\partial \theta_j} (z(\xi_i) - \tilde{v}) \mathbf{1}_{z(\xi_i) \leq \tilde{v}}$.
Return $\Delta_W$.
```

can be written as [12]

$$\frac{\partial \log f_\Xi(\xi;\theta)}{\partial \theta_j} = \sum_{k=0}^{\tau-1} \frac{\log \Pr(u_k \mid x_k;\theta)}{\partial \theta_j}, \tag{10}$$

where $\zeta_0(x_0)$ is the probability distribution of the initial condition x_0 and τ is the total number of time steps considered. We can use formulas (10) to calculate $\frac{\partial \log f_\Xi(\xi_i;\theta)}{\partial \theta_j}$ in (9).

To learn the gradient (9), in the learning phase, given the initial policy parameter θ_0, the robot starts with a fixed initial position x_0 in the grid world and moves in the workspace based on the parameter θ_0. The robot repeats the process W times and generates a sample of W trajectories $\{\xi_1, ..., \xi_W\}$. After the robot has recorded the sample, it executes Algorithm 1 to calculate the gradient estimate.

The output of Algorithm 1 is the gradient estimate Δ_W. The robot uses gradient ascent to update the parameter θ based on the gradient estimate Δ_W as

$$\theta_{l+1} = \theta_l + \epsilon_l \Delta_W, \tag{11}$$

where ϵ_l is a step size. Note that the subscript l in (11) is the iteration index of the gradient ascent algorithm. Each iterate θ_l is an M-dimensional vector. Denote by $\theta_{j,l}$ the j-th element of θ_l.

It is shown in [12] that the gradient estimator Δ_W is a biased and consistent estimator of $\nabla \phi_\beta(Z;\theta)$.

Proposition 2: [12] If Assumptions (1)-(4) hold, then for $j = 1, ..., M$, $\Delta_{j,W} \to \frac{\partial \phi_\beta(Z;\theta)}{\partial \theta_j}$ w.p. 1 as $W \to \infty$.

Similar to Section III-A, under Assumptions 1-4, the update (11) will lead the parameter θ to a locally optimal point [12].

IV. Distributed Risk-Averse Policy Gradient

In the previous section, we considered a single robot that explores the workspace and learns the process along the way. In the learning phase (Algorithm 1), there are multiple trajectories required to update the policy parameter, so the workload for the single robot is high. In this section, we propose a way to parallelize the learning task by considering multiple robots that collaboratively explore the environment.

The key idea for our distributed learning framework follows the approach proposed in [5]. In particular, we assume that each robot i has its own parameter θ^i and generates its own gradient estimate $\Delta^i_{W^i}$ with sequences of realizations $\xi^i = (x^i_0, u^i_0, r^i_0, ..., x^i_K, u^i_K, r^i_K)$. Then, the policy of every robot i is (cf. (2) and (3))

$$\mu_{\theta^i}(u^i \mid x^i_k) = \frac{\exp(\eta_{\theta^i}(u^i, x^i_k))}{\sum_{v \in \mathcal{U}} \exp(\eta_{\theta^i}(v^i, x^i_k))}, \tag{12}$$

where

$$\eta_{\theta^i}(u^i, x^i) = \sum_{j=1}^{M} \theta^i_j s^j_k(x^i + u^i). \tag{13}$$

and the parameters θ^i are local to every robot. The goal is that all robots coordinate to estimate and update a common parameter θ in a distributed way. For this, we introduce a consensus update to ensure that the policy parameters θ^i for all $i \in \mathcal{N}$ converge to the same value θ. Agreement on the parameter θ means that the robots will collaboratively learn a common policy parameterized by θ. Specifically, consider a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, over which the robots communicate in order to update θ^i. The nodes in the graph $\mathcal{G}$ correspond to the robots, while arcs denote communication links between robots, so that, if robot i and robot j can communicate with each other, then T^{ij} is the time that robot i receives a message from robot j and $t^{ij}(k)$ is the time this message was sent.

Using the above formulation, we can design a distributed policy gradient algorithm where, in the first step of the algorithm, each robot carries out its own gradient estimation (Algorithm 1) and in the second step of the algorithm, each robot updates its parameter θ^i by

$$\theta^i_{l+1} = A^{ii}_l \theta^i_l + \sum_{j \neq i} A^{ij}_l \theta^j_{t^{ij}(l)} + \epsilon_l \Delta^i_W, \tag{14}$$

where $A^{ij}_l = \text{diag}(a^{ij}_{1,l}, ..., a^{ij}_{M,l})$, $A^{ij}_l \geq 0$, $\sum_{j=1}^{N} a^{ij}_{m,l} = 1$ for all i, m, l, and $A^{ij}_l = 0$ if $l \notin T^{ij}$ for all $i \neq j$. Each robot's parameter update in (14) consists of two parts: the consensus update $A^{ii}_l \theta^i_l + \sum_{j \neq i} A^{ij}_l \theta^j_{t^{ij}(l)}$ and the update in the gradient direction $\epsilon_l \Delta^i_W$.

The proposed distributed policy gradient method relies on the following additional assumptions on the graph $\mathcal{G}$ and communication pattern.

Assumption 5: Consider a communication graph as defined above. Then:
(1) The graph $\mathcal{G}$ is strongly connected, namely, there exists a directed path in $\mathcal{G}$ from every node to every other node.
(2) There exists a positive constant γ such that $A^{ii}_l \geq \gamma \mathbf{I}$ for all $i \in \mathcal{N}$ and l, and $A^{ij}_l \geq \gamma \mathbf{I}$ or all $i, j \in \mathcal{N}$ and l.
(3) The time between consecutive transmissions of θ^i_l from agent j to agent i is bounded by some $B \geq 0$ for all $(j, i) \in \mathcal{E}$.
(4) Communication delays are bounded by some $B_0 \geq 0$.

Assumption 5 says that information can flow between any two agents via (14); the weights of the θ's are bounded away from zero; and for every robot i, there is a finite delay between two consecutive receptions of a message θ^j originating from a neighboring robot j.

By defining $s_l^i = \Delta_W^i$, we can rewrite (14) as follows:

$$\theta_{l+1}^i = A_l^{ii}\theta_l^i + \sum_{j\neq i} A_l^{ij}\theta_{t^{ij}(l)}^j + \epsilon_l s_l^i, \tag{15}$$

where the term s_l^i can be viewed as a noise term which perturbs the consensus update in (15); see [5]. Denote by y_l the value of θ that all robots would agree on if at time l they employ a different parameter update that involves only the consensus term (the first two terms in (15)). As shown in [13], the update (15) has the following properties.

Lemma 1: [13] Under Assumption 5, the update (15) has the following properties:
(1) There exist vectors $\Phi_{n,l}^{ij}$ for all $l \geq n$ and $i,\ j \in \mathcal{N}$ such that

$$\theta_l^i = \sum_{j=1}^N \Phi_{0,l}^{ij}\theta_1^j + \sum_{n=1}^{l-1}\epsilon_n \Phi_{n,l}^{ij} s_n^j, \tag{16}$$

and the limit $\lim_{l\to\infty}\Phi_{n,l}^{ij}$ exists, for any i, j, n. The limit is independent of i and will be denoted by Φ_n^j.
(2) There exist $d \in [0,1]$ and $\tilde{A} \geq 0$ such that

$$\max_{i,j}\|\Phi_{n,l}^{ij} - \Phi_n^j\| \leq \tilde{A} d^{l-n}, \quad l \geq n,\ i,\ j \in \mathcal{N}. \tag{17}$$

(3) The vector y_l can be obtained by the update

$$y_l = \sum_{j=1}^N \Phi_0^j\theta_1^j + \sum_{n=1}^{l-1}\epsilon_n\Phi_n^j s_n^j, \quad l \geq 0. \tag{18}$$

In general, the objective $\phi_\beta(\theta)$ is not concave in θ. In the following, we start with an assumption on the step size ϵ_l in the update (14) and show that the update (14) will lead the parameter θ^i to a locally optimal solution. Moreover, consensus on the parameter will be reached, i.e., $\|\theta_l^i - \theta_l^j\| \to 0$ as $l \to \infty$, for all $i,\ j \in \mathcal{N}$.

Assumption 6: The step size ϵ_l satisfies

$$\sum_l \epsilon_l = \infty, \quad \sum_l (\epsilon_l)^2 < \infty. \tag{19}$$

Theorem 1: Let Assumptions (1)-(6) hold. Also, assume that $\Delta_{j,W_l} \to \frac{\partial\phi_\beta(\theta_l^i)}{\partial\theta_j}$ w.p. 1 for all $i \in \mathcal{N}$, and $j \in \mathcal{M}$. Then the sequences $\{\theta_l^i\}$ generated by the algorithm satisfy

$$\liminf_{l\to\infty}\|\nabla\phi_\beta(\theta_l^i)\| = 0,\ \forall i \in \mathcal{N},\ \text{w.p. 1}. \tag{20}$$

Proof: Each robot's update (14) can be written as

$$\begin{aligned}\theta_{l+1}^i &= A_l^{ii}\theta_l^i + \sum_{j\neq i}A_l^{ij}\theta_{t^{ij}(l)}^j + \epsilon_l\Delta_W^i \\ &= \theta_l^i + \epsilon_l\nabla\phi_\beta(\theta_l^i) + e_{1,l}^i + e_{2,l}^i,\end{aligned} \tag{21}$$

where $e_{1,l}^i = \epsilon_l\big(\Delta_W^i - \nabla\phi_\beta(\theta_l^i)\big)$ and $e_{2,l}^i = A_l^{ii}\theta_l^i + \sum_{j\neq i}A_l^{ij}\theta_{t^{ij}(l)}^j - \theta_l^i$.

Following the analysis in [5] and [13], we can show that $e_{2,j} \to 1$ w.p. 1. Specifically, let,

$$b_l = \sum_{i=1}^N \epsilon_l\|\Delta_W^i\|. \tag{22}$$

Using Lemma 1, we can bound the difference between the sequence θ_l^i and y_l. In particular, we have that there exist $d \in [0,1)$ and $A \geq 0$ such that

$$\begin{aligned}\|y_l - \theta_l^i\| &\leq A\sum_{n=1}^l d^{l-n}b_n \\ &= Ab_1\left(d^{(l-1)} + \sum_{n=1}^{l-1}\frac{d^{l-n-1}b_{n+1}}{b_1}\right)\end{aligned} \tag{23}$$

Furthermore, we have

$$\begin{aligned}\mathbb{E}\Big[\sum_l (b_l)^2\Big] &\leq \mathbb{E}\Big[\sum_l N\sum_{i=1}^N \epsilon_l^2\|\Delta_W^i\|^2\Big] \\ &\leq N\sum_l(\epsilon_l)^2\sum_{i=1}^N\mathbb{E}[\|\Delta_W^i\|^2] \leq \infty,\end{aligned} \tag{24}$$

where the second inequality is due to the boundedness assumptions on z and $\frac{\partial\log f_\Xi(\xi;\theta)}{\partial\theta_j}$. Hence, b_l converges to zero almost surely. Due to (23), $y_l - \theta_l^i$ converges to zero for all $i \in \mathcal{N}$ almost surely, and e_2^1 converges to zero almost surely as well.

The convergence of the error term $e_{1,l}^i$ can be established by the consistency property (Proposition 2 in Section III-B), according to which, $\Delta_{j,W_l} \to \frac{\partial\phi_\beta(\theta_l^i)}{\partial\theta_j}$ w.p. 1, as $W_l \to \infty$ for all $i \in \mathcal{N}$ and $j = 1, ..., M$.

Following the analysis in Theorem 6.3 in [4] and using the Taylor's series, we have

$$\begin{aligned}\phi_\beta(\theta_{l+1}^i) \geq \phi_\beta(\theta_l^i) &+ \nabla\phi_\beta(\theta_l^i)^\top(\epsilon_l\nabla\phi_\beta(\theta_l^i) + e_{1,l}^i + e_{2,l}^i) \\ &+ \tilde{C}\cdot\|\epsilon_l\nabla\phi_\beta(\theta_l^i) + e_{1,l}^i + e_{2,l}^i\|^2,\end{aligned} \tag{25}$$

where $\tilde{C}$ reflects a lower bound on the Hessian of ϕ_β. For some $T > 0$, define $l_{j+1} = \min\{l \geq l_j \mid \sum_{t=l_j}^l \epsilon_t \geq T\}$ for $j > 0$. Using (25), we have

$$\begin{aligned}\phi_\beta(\theta_{l_{j+1}}^i) \geq \phi_\beta(\theta_{l_j}^i) &+ \sum_{t=l_j}^{l_{j+1}-1}\epsilon_t\|\nabla\phi_\beta(\theta_t^i)\|^2 \\ &+ \sum_{t=l_j}^{l_{j+1}-1}\nabla\phi_\beta(\theta_t^i)(e_{1,t}^i + e_{2,t}^i) \\ &+ \sum_{t=l_j}^{l_{j+1}-1}\tilde{C}\cdot\|\epsilon_t\nabla\phi_\beta(\theta_t^i) + e_{1,t}^i + e_{2,t}^i\|^2,\end{aligned} \tag{26}$$

where the last two terms in (26) converge to zero. Now assume that $\|\nabla\phi_\beta(\theta_t^i)\|$ does not converge to zero. Then $\phi_\beta(\theta_l^i)$ will increase indefinitely, which contradicts the boundedness assumption on ϕ_β. This completes the proof. ■

Theorem 1 shows that all robots agree on the same policy parameter which converges to a locally optimal solution. The difference between the proof of Theorem 1 and the analysis in [5] lies in the noise term s_l^i in (15) derived from the policy gradient algorithm which has a different structure compared to the noise term in [5]. Note that the requirement $\Delta_{j,W_l} \to \frac{\partial\phi_\beta(\theta_l^i)}{\partial\theta_j}$ w.p. 1 for all $i \in \mathcal{N}$ indicates that the number of MC samples $W_l \to \infty$ as $l \to \infty$.

V. Numerical Simulations

In this section we illustrate our distributed risk-averse policy gradient method by comparing it to the risk-neutral case. We construct a workspace using a grid world as in Fig. 1. In the first example, we use only one robot to explore the workspace. In the workspace, there are two static sources located at positions $(9, 3)$ and $(3, 9)$, respectively. We partition the workspace into two regions. The grey and white regions are defined as the risky and safe regions, respectively. A risky region provides larger average rewards with higher variance than the safe one. In particular, each position in the risky region has random reward $200 \times a + d$, where a has a uniform distribution over $[0, 1]$ and $d \sim \mathcal{N}(0, 10^4)$. Source 2 inside the risky region has a reward $v_k^2 \sim \mathcal{N}(2000, 4 \times 10^6)$ for all $k \in \mathcal{K}$. Each reward in the risky region is bounded in the interval $[-10^4, 10^4]$. Each position in the safe region has random reward $100 \times a$. Source 1 that is in the safe region has a reward $v_k^1 = 1000$ for all $k \in \mathcal{K}$. The two sources in the workspace generate the signal based on a two-dimensional Gaussian distribution described in (1).

We compare the risk-neutral approach (Section III-A) that determines the final policy parameterized by the parameter θ that optimizes the average cumulative rewards with the risk-averse approach (Section III-B) that optimizes the CVaR of the cumulative rewards. We use 6000 MC samples ($W = 6000$) for both the risk-neutral and risk-averse approaches. Fig. 2 shows that the risk-averse approach results in larger CVaR of the total reward than the risk-neutral one. In Fig. 1, we plot trajectories obtained by the risk-neutral and risk-averse policies. The grey and white regions are the risky and safe regions, respectively. Red stars represent the targets and squares are the final positions of the robots. Blue and black lines represent realizations of the trajectories obtained by the risk-neutral and risk-averse policy, respectively. Unlike the risk-neutral one, the risk-averse final policy generates a sample path which only marginally extends to the risky region.

In the second example, we verify the convergence of the distributed policy gradient algorithm proposed in Section IV. We consider three robots that carry their own gradient estimation. Robot 1 can communicate with Robot 2; Robot 2 can communicate with Robot 1 and 3; and Robot 3 can communicate with Robot 2. We compare a centralized solution, where a single robot uses 6000 MC samples ($W = 6000$) in the policy gradient algorithm with a distributed solution where each one of the three robots uses 2000 MC samples ($W_i = 2000$) so that the task is run in parallel. Fig. 3 shows the average and CVaR of the total rewards obtained by the centralized and distributed approaches. We can see that both the average and CVaR of the total rewards obtained by the distributed approach converge to their corresponding centralized values. Fig. 4 shows the convergence of the consensus. Specifically, we plot $\max_{i,j} \|\theta_l^i - \theta_l^j\|$ versus the number of iterations used in the update (15). We can see that each agent's parameter θ_l^i converges to the same value and reaches consensus on the final policy.

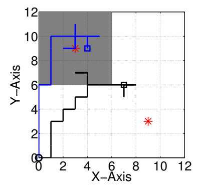

Fig. 1. Configuration of the grid world and the trajectories of the robots.

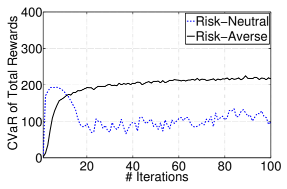

Fig. 2. CVaR of total rewards obtained by the risk-neutral and risk-averse approaches.

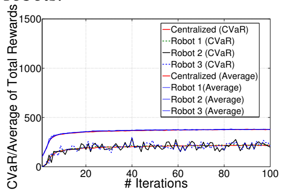

Fig. 3. Convergence results of the average and CVaR of the total rewards of the consensus algorithm.

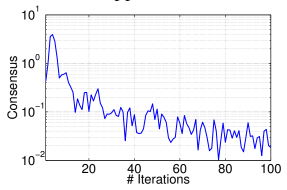

Fig. 4. Convergence result of the consensus using distributed policy gradient algorithm.

VI. Conclusions

We presented a distributed risk-averse reinforcement learning approach to planning the motion of a network of sensors tasked with jointly sensing an unknown environment. The proposed risk-averse method provides a solution, which is less conservative than worst-case approaches and takes into account the extreme low-reward events by shaping the tail of the probability distribution of the random decisions.

References

[1] D. P. Bertsekas, "Dynamic programming and optimal control," vol. 1, 3rd ed., Athena Scientific, 2005.

[2] Y. Chow and M. Ghavamzadeh, "Algorithms for cvar optimization in mdps," In *NIPS* 27, 2014.

[3] Y. Chow and M. Pavone, "A unifying framework for time-consistent, risk-averse model predictive control: theory and algorithms," In *American Control Conference*, 2014.

[4] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," SIAM J. Control and Optimization, vol. 42, no. 4, pp. 1143-1166, 2003.

[5] P. Pennesi and I. C. Paschalidis, "A distributed actor-critic algorithm and applications to mobile sensor network coordination problems," in *IEEE* Transactions on Automatic Control, vol. 55, no. 2, pp. 492-497, Feb. 2010.

[6] R. Rockafellar and S. Uryasev, "Conditional value-at-risk for general loss distributions," Journal of Banking & Finance, vol. 26, no. 7, pp. 1443-1471, 2002.

[7] A. Ruszczynski, "Risk-averse dynamic programming for markov decision processes," *Mathematical Programming*, 125(2):235-261, 2010.

[8] A. Shapiro, D. Dentcheva, and A. Ruszczynski, "Lectures on stochastic programming," SIAM, 2009.

[9] R. Sutton and A. Barto, "Reinforcement learning: an introduction," Cambridge Univ. Press, 1998.

[10] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," In *NIPS* 13, 2000.

[11] A. Tamar, Y. Chow, M. Ghavamzadeh and S. Mannor, "Policy gradient for coherent risk measures," In *NIPS* 28, 2015.

[12] A. Tamar, Y. Glassner, and S. Mannor, "Policy gradients beyond expectations: conditional value-at-risk," arXiv:1404.3862v4, 2014.

[13] J. N. Tsitsiklis, D. P. Bertsekas and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," in *IEEE Transactions on Automatic Control*, AC-31(9): 803 - 812, 1986.