

# The Price of Decentralization: Event-Driven Optimization for Multi-Agent Persistent Monitoring Tasks

Nan Zhou<sup>1</sup>, Christos G. Cassandras<sup>1,2</sup>, Xi Yu<sup>3</sup>, and Sean B. Andersson<sup>1,3</sup>

**Abstract**—In persistent monitoring tasks, the objective is to control the movements of cooperating agents in order to minimize an uncertainty metric associated with a finite number of targets. We formulate an optimal control problem and show that the optimal solution can be reduced to or approximated by parametric agent trajectory families. The behavior of agents and targets under optimal control can be described by a hybrid system. This enables the use of Infinitesimal Perturbation Analysis (IPA) to obtain an on-line *centralized* solution through a gradient-based algorithm. We identify conditions under which this centralized solution to the parametric optimization problems can be recovered in a *decentralized* and event-driven manner. In the decentralized scheme, each agent optimizes its performance based on local information, except for one type of non-local event requiring communication from a non-neighbor agent, giving rise to a quantifiable “price of decentralization”. Simulation examples are included to illustrate the effectiveness of this “almost decentralized” optimization algorithm and compare it to its fully decentralized counterpart where the aforementioned non-local event is ignored.

## I. INTRODUCTION

A cooperative multi-agent system consists of interacting agents, each controlling its local state so as to collectively optimize a common global objective subject to various constraints. Such systems have seen significant success in performing a variety of collaborative tasks such as environmental sensing, sampling, coverage, and surveillance [1]–[3]. Many of these tasks can be formulated as static optimization problems, whereby agents are assigned to real-valued locations in a given space. In contrast, *persistent monitoring* is a task that involves dynamic optimization, i.e., the control of agent movement over time so that agents can cooperatively monitor a dynamically changing environment that cannot be fully covered by a stationary team of agents (as in coverage control) [4]–[6]. Unlike sweep coverage [2] and patrolling tasks [7]–[9] where *every* point in a mission space is of interest, the problem we address here focuses on a *finite number* of data sources or “targets” (typically larger than the number of agents).

In this setting, the agents interact with targets through their sensing capabilities which are normally dependent upon their physical distance from a target. Applications of this setting include the monitoring of intersections in a transportation network where agents are vehicles moving along the links of

this network to collect information at each such intersection; the sampling of ocean temperature where agents are deployed to periodically measure the temperature at multiple targeted locations; and the tracking of bio-molecular evolution where agents are microscopic beams controlled to observe the behavior of molecules at multiple targets. The underlying problem we consider falls within the large class of resource allocation and scheduling problems often viewed as applications of the Traveling Salesman Problem (TSP) or its extension to Multiple traveling salesmen (MTSP). Thus, the entire range of TSP applications overlaps with the persistent monitoring problem which is in fact much more complicated than the well-known NP-Hard TSP due to: (i) the presence of multiple agents, (ii) the need to determine dwell times at each visited target, (iii) the target dynamics and (iv) the freedom to make multiple visits to targets. The same reasons also make it computationally intractable to apply dynamic programming techniques so as to obtain the optimal controls, even for a relatively simple problem.

Unlike many multi-agent systems composed solely of a network of interconnected agents, this agent-target interaction is modeled through a network whose nodes consist of *both* agents and targets. The purpose of the agents in persistent monitoring is to cooperatively estimate the state of the targets with maximal accuracy or, equivalently, to minimize a measure of target state uncertainty. Since agents are mobile, the overall graph topology in such systems is time-varying, thus exacerbating the complexity of this class of problems. This has motivated approaches where, rather than viewing these as discrete agent-to-target assignment problems [10], [11] (which are computationally intensive and do not scale well), one treats them as trajectory design dynamic optimization problems [12], [13]. The solutions obtained for these problems have thus far been centralized [1], [9], [12], [14], [15], which renders them energy-inefficient due to excessive communication, as well as unreliable, especially in adversarial environments [16]. While distributed algorithms have been derived and applied to coverage control, formation control, and consensus problems [17], [18], decentralization in a persistent monitoring setting is particularly challenging due to the time-varying nature of the agent-target network and the fact that agents take actions depending on interactions with the environment (targets) which cannot be easily shared through the agent network.

In this paper, we address the question of whether it is possible to develop *decentralized* controllers for multi-agent dynamic optimization and, in particular, persistent monitoring problems. More precisely, we view decentralization as a process which aims to achieve *the same performance as a central controller* by distributing functionality to the agents so that

The work of Cassandras and Zhou is supported in part by NSF under grants ECCS-1509084, DMS-1664644, and CNS1645681, by AFOSR under grant FA9550-19-1-0158, by ARPA-Es NEXTCAR program under grant DE-AR0000796, and by the MathWorks. The work of Andersson and Yu is supported in part by the NSF through grant ECCS-1509084.

The authors are with <sup>1</sup>Division of Systems Engineering, <sup>2</sup>Department of Electrical and Computer Engineering, and <sup>3</sup>Department of Mechanical Engineering, Boston University, Boston, MA, USA. (email: {nanzhou, cgc, xyu, sanderss}@bu.edu)

each one acts based on local information or by communicating with only a set of neighbors.

Our analysis builds on solutions of the one-dimensional (1D) persistent monitoring problem which were shown [12] to be parametric, i.e., the underlying optimal control problem is reduced to one involving parametric optimization. In particular, the optimal solution is characterized by a finite number of points where an agent switches direction, along with a dwell time at each such point. In two-dimensional (2D) spaces, such parametric representations for optimal agent trajectories have been shown to no longer hold [9]. Nonetheless, by considering parametric families of agent trajectories (e.g., ellipses, Lissajous and Fourier curves, or interconnected linear segments) optimal trajectories within these families can be obtained [9]. Agent-target interactions under the aforementioned parametric controllers can be described by a hybrid dynamic system. This enables the use of Infinitesimal Perturbation Analysis (IPA) [19], [20] to determine on-line optimal parameters through event-driven gradient descent algorithms. This approach exploits the event-driven nature of IPA to render it *scalable* in the number of *events* (all of which are explicitly defined in Section V) in the system and not its state space. Moreover, we will show that the question of decentralization can be conveniently reduced to one of *event observability*, i.e., whether an agent can observe all the events it requires to evaluate its local IPA gradient.

The contribution of this paper consists of identifying explicit conditions under which the centralized solutions to the optimal persistent monitoring problems (e.g., the ones studied in [9], [12]) can be recovered without any performance degradation. We will show that this is possible through “almost decentralized” and entirely event-driven agent controllers. In contrast to the previous 1D results in [21], this paper formulates the problem in 2D, extends the IPA gradient-based algorithm, and proves that conditions of the decentralization solution continue to hold in 2D. In particular, each agent uses (i) its own local information (to be precisely defined), (ii) information in the form of observable events from agents that happen to be its neighbors at the time such events occur, and (iii) a single specific event type communicated by a non-neighbor agent when it occurs. It is the latter that prevents a completely decentralized control scheme and incurs a “price of decentralization”. As we observe in simulations, ignoring such non-local events often results in little loss of accuracy. In addition, we develop such an “almost decentralized” algorithm which, compared to the centralized solutions, significantly reduces the information required for each agent in order to solve the problem while yielding the same performance.

The paper is organized as follows. Section II formulates persistent monitoring as an optimal control problem. Section III presents a Hamiltonian analysis which characterizes the optimal solution in terms of specifying a trajectory for each agent. Section IV introduces the limited information model needed for decentralization and Section V describes event-driven IPA gradient estimation. In Section VI we present our main results regarding the “price of decentralization” along with a decentralized event-driven algorithm using IPA gradient descent. Section VII provides simulation examples to illustrate

the proposed decentralization scheme. Section VIII concludes the paper.

## II. PROBLEM FORMULATION

We begin by generalizing the 1D persistent monitoring problem formulation in [12] to a 2D setting.

**Agent dynamics.** We consider a team of  $N$  agents (indexed by  $j$ ) operating in a 2D mission space  $\Omega \subseteq \mathbb{R}^2$ . The position of an agent is denoted by  $s_j(t)$  and follows the dynamics:

$$\dot{s}_j(t) = u_j(t) \begin{bmatrix} \cos \gamma_j(t) \\ \sin \gamma_j(t) \end{bmatrix} \quad (1)$$

where  $u_j(t)$  is the speed and  $\gamma_j(t)$  the direction. Without loss of generality, the control input is scaled and bounded such that  $\|u_j(t)\| \leq 1$  and  $\gamma_j(t) \in [0, \pi)$ . The initial state  $s_j(0)$  of every agent,  $j = 1, \dots, N$ , is assumed given and we do not constrain the final state as long as the agent performance is optimized.

**Agent sensing model.** In the persistent monitoring setting, an agent senses the environment and detects events in its vicinity. This detection/sensing ability is modeled by a function  $p_j(x, s_j(t))$  that measures the probability that an event at location  $x \in \Omega$  is detected by agent  $j$ . We assume that  $p_j(x, s_j) = 1$  if  $x = s_j$ , and that  $p_j(x, s_j(t))$  is monotonically non-increasing in the Euclidean distance  $\|x - s_j(t)\|$ , thus capturing the reduced effectiveness of a sensor over its range. We consider this range to be finite and denoted by  $r_j$ . Although our analysis is not affected by the precise sensing model  $p_j(x, s_j(t))$ , we will limit ourselves for simplicity to a linear decay model as follows:

$$p_j(x, s_j(t)) = \max \left\{ 0, 1 - \frac{\|x - s_j(t)\|}{r_j} \right\} \quad (2)$$

Since we are interested in how an agent  $j$  senses a target  $i$  located at  $x_i$ , we set  $p_j(x_i, s_j(t)) \equiv p_{ij}(s_j(t))$ . For  $N$  agents sensing cooperatively, assuming detection independence, the joint probability that target  $i$  is sensed by at least one agent is captured by

$$P_i(s(t)) = 1 - \prod_{j=1}^N (1 - p_{ij}(s_j(t))) \quad (3)$$

where we set  $s(t) = [s_1(t), \dots, s_N(t)]^\top$ .

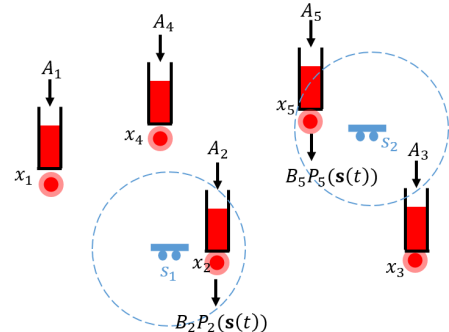


Fig. 1: A queueing model interpretation for P1.

**Target model.** We consider a finite set of targets at fixed locations  $x_i \in \Omega$ ,  $i = 1, \dots, M$ . We associate with each target  $i$  a state  $R_i(t)$ . Since agents interact with targets, this state is affected by the agent states  $\mathbf{s}(t)$ . Its dynamics have the form:

$$\dot{R}_i(t) = \begin{cases} 0 & \text{if } R_i(t) = 0 \text{ and } A_i \leq B_i P_i(\mathbf{s}(t)) \\ A_i - B_i P_i(\mathbf{s}(t)) & \text{otherwise} \end{cases} \quad (4)$$

with a given initial condition  $R_i(0)$ ,  $i = 1, \dots, M$  and  $B_i > A_i > 0$  to ensure a strict decrease in  $R_i(t)$  when  $P_i(\mathbf{s}(t)) = 1$ . The target value grows linearly when there is no agent within its vicinity and decays in proportion to the agent's detection probability. In this case,  $R_i(t)$  measures the *uncertainty state* of target  $i$  and the only other available information about it is its location  $x_i \in \Omega$ . This model has an intuitive queueing system interpretation as shown in Fig. 1, i.e.,  $R_i(t)$  is the time-varying ‘‘content’’ of a queue (representing the uncertainty about the knowledge of the target), with inflow rate  $A_i$  and agent-dependent outflow rate  $B_i P_i(\mathbf{s}(t))$ . This model applies, for example, to data centers where  $R_i(t)$  captures the accumulated information requests to be processed, or, in traffic networks where different locations (e.g., intersections) need to be persistently monitored.

Under the model (4), our goal is to control the movement of agents through  $\mathbf{u}(t) = [u_1(t), \dots, u_N(t)]^\top$  and  $\gamma(t) = [\gamma_1(t), \dots, \gamma_N(t)]^\top$  so that the cumulative average uncertainty over all targets  $i = 1, \dots, M$  is minimized over a finite time horizon  $T$ . The associated optimal control problem is stated as follows:

$$\mathbf{P1} : \min_{\mathbf{u}(t), \gamma(t)} J = \frac{1}{T} \int_0^T \sum_{i=1}^M R_i(t) dt \quad (5)$$

subject to the agent dynamics (1), target uncertainty dynamics (4), and the sensing model (2) and (3).

### III. PROPERTIES OF THE OPTIMAL CONTROL SOLUTION

The 1D version of **P1** was explicitly studied in [12]. Here, we extend the Hamiltonian analysis to the 2D version. Define the state vector  $\mathbf{x}(t) = [R_1(t), \dots, R_M(t), s_1^x(t), s_1^y(t), \dots, s_N^x(t), s_N^y(t)]$  and associated costate vector  $\lambda = [\lambda_1(t), \dots, \lambda_M(t), \lambda_{s_1}^x(t), \lambda_{s_1}^y(t), \dots, \lambda_{s_N}^x(t), \lambda_{s_N}^y(t)]$ . Due to the discontinuity in the dynamics of  $R_i(t)$  in (4), the optimal state trajectory may contain a boundary arc over which  $R_i(t) = 0$  for some  $i$ ; otherwise, the state evolves in an interior arc. Using (1) and (4), the Hamiltonian is

$$H(\mathbf{x}, \lambda, \mathbf{u}, \gamma) = \sum_{i=1}^M R_i(t) + \sum_{i=1}^M \lambda_i(t) \dot{R}_i(t) + \sum_{j=1}^N u_j(t) \left( \lambda_{s_j}^x(t) \cos \gamma_j(t) + \lambda_{s_j}^y(t) \sin \gamma_j(t) \right) \quad (6)$$

and a straightforward application of the Pontryagin Minimum Principle implies that

$$\|u_j^*(t)\| = \begin{cases} 1 & \text{if } \lambda_{s_j}^x(t) \cos \gamma_j(t) + \lambda_{s_j}^y(t) \sin \gamma_j(t) \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Thus, each agent moves with maximum speed or dwells in singular arcs that may exist. The detailed analysis is similar to the one presented in [9], [12] and is, therefore, omitted here.

A complete solution of the optimal control problem **P1** involves solving a computationally intensive Two Point Boundary Value Problem (TPBVP) which is generally intractable and requires global information of all agents and targets. However, based on the fact that  $\|u_j^*(t)\| \in \{1, 0\}$ , it is clear that the optimal solution can be reduced to determining (i) the agent's heading  $\gamma_j^*(t)$  and (ii) possible dwell times when the agent switches its control from  $\|u_j^*(t)\| = 1$  to 0. This motivates us to explore solutions constrained to parameterized trajectory families. In fact, in the 1D case studied in [12] the optimal solution is indeed parametric. In the 2D case, we consider trajectory families which may be parameterized according to desirable properties and then seek to optimize the trajectories within such families. Such desirable properties include periodicity (so that every target is guaranteed to be visited within the given time interval upper bound) and smoothness (in applications where the agents are subject to specific motion constraints). Therefore, each agent trajectory  $s_j(\theta_j, t, s_{j,0})$  may be constrained to a specific parametric form characterized by a vector  $\theta_j \in \mathbb{R}^p$  for a trajectory family fully specified by  $p$  parameters (e.g., elliptical trajectories where  $p = 5$  which are considered in Section VII-B). Thus, in general, we denote by  $J(\theta)$  the parametric form of the objective function, so that **P1** becomes

$$\min_{\theta} J(\theta) = \frac{1}{T} \int_0^T \sum_{i=1}^M R_i(t; \theta) dt \quad (8)$$

where  $\theta$  is a parameter vector that fully characterizes the agent trajectory family of interest (e.g., the parameters that describe a family of ellipses or Fourier curves).

Under such parameterizations, the agent and target dynamics in (1) and (4) define a *hybrid system*. Letting  $\tau_k(\theta)$  denote the occurrence time of the  $k$ -th event that switches the agent or target dynamics in this hybrid system operating under a parameter  $\theta$ , we can rewrite (8) as

$$\min_{\theta} J(\theta) = \frac{1}{T} \sum_{k=0}^K \int_{\tau_k(\theta)}^{\tau_{k+1}(\theta)} \sum_{i=1}^M R_i(t; \theta) dt \quad (9)$$

where  $K$  denotes the total number of events. This problem can be solved using standard gradient-based algorithms which rely on the gradient  $\nabla J(\theta)$ . The hybrid nature of this system allows us to use Infinitesimal Perturbation Analysis (IPA) [19] to determine the gradient online in an event-driven fashion. Then we can optimize the objective function by selecting an initial parameter setting  $\theta_0$  and iteratively adjusting its value through a gradient-based algorithm, as further described in Section VI.

Thus far, solutions to such persistent monitoring problems have been developed in a purely centralized manner, e.g., [12]. The rest of this paper explores the possibility for decentralizing such solutions, i.e., limiting each agent to information that is only locally available while maintaining the same performance as a centralized controller. If that is not always possible, our goal is to determine conditions under which it is and quantify

the “price of decentralization” in the sense of performance degradation that results from decentralized solutions to problem (9).

#### IV. LIMITED INFORMATION MODEL FOR DECENTRALIZATION

In order to properly define a decentralized persistent monitoring problem, we first review the limited information model introduced in [21]. In our model, an agent is capable of observing information within its sensing range, specifically the target state  $R_i(t)$  when  $\|x_i - s_j(t)\| \leq r_j$ . We restrict the observations to  $r_j$ , since  $p_{ij}(s_j(t)) = 0$  outside this range as in (2). Moreover, each agent can communicate with a set of “neighboring” agents to acquire information such as their positions, speeds, and the states of targets which are within these neighbors’ sensing ranges. In contrast to traditional multi-agent systems modeled exclusively through a network of agents, the persistent monitoring network includes both agents and targets. Accordingly, we revisit the concept of “neighborhood” and account for the fact that neighborhoods are time-varying.

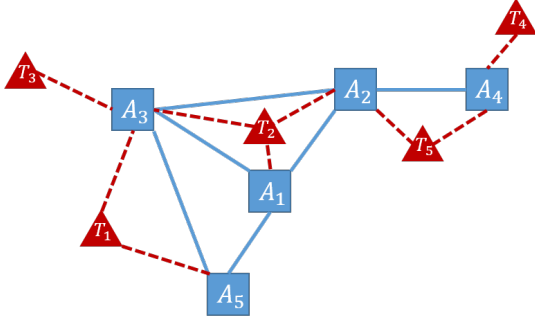


Fig. 2: Agent-target network. Red triangles are targets and blue squares are agents. Blue lines indicate the connections of neighboring agents of an agent and red lines indicate the connections of neighboring targets of an agent.

Agents have two types of neighbors: nearby agents and nearby targets.

**Definition 1.** *The agent neighborhood of agent  $j$  is the set  $\mathcal{A}_j(t) = \{k : \|s_k(t) - s_j(t)\| \leq r_c, k \neq j, k = 1, \dots, N\}$ .*

This is a conventional definition of neighbors in multi-agent systems, where  $r_c$  is a communication range, but we point out that it is *time-dependent* since agents are generally moving. As an example, in Fig. 2,  $\mathcal{A}_1 = \{A_2, A_3, A_5\}$ .

**Definition 2.** *The target neighborhood of agent  $j$  is the set  $\mathcal{T}_j(t) = \{i : \|x_i - s_j(t)\| \leq r_j, i = 1, \dots, M\}$ .*

This includes all targets which are within agent  $j$ ’s sensing range. In the example of Fig. 2,  $\mathcal{T}_3 = \{T_1, T_2, T_3\}$ . Assuming the agents are homogeneous with a common sensing range  $r$ , we require that  $r_c \geq 2r$  in order to establish communication among agents that are sensing the same target. On the other hand, the neighborhood of a target consists only of nearby agents (we do not explicitly model any possible connectivity among targets).

**Definition 3.** *The agent neighborhood of target  $i$  is the set  $\mathcal{B}_i(t) = \{j : \|s_j(t) - x_i\| \leq r_j, j = 1, \dots, N\}$ .*

This set captures all the neighbor agents of target  $i$ . In the example of Fig. 2,  $\mathcal{B}_2 = \{A_1, A_2, A_3\}$ .

We further define

$$\mathcal{N}_{ij}(t) = \mathcal{B}_i(t) \setminus \{j\} \quad (10)$$

to indicate the “collaborators” of agent  $j$  in sensing target  $i$ . Note that  $\mathcal{N}_{ij}(t) = \{k : k \in \mathcal{A}_j(t) \text{ and } k \in \mathcal{B}_i(t)\}$ , thus capturing a neighbor of agent  $j$  and target  $i$  at the same time.

Using (10), the joint sensing probability in (3) can be rewritten as:

$$P_i(s(t)) = 1 - (1 - p_{ij}(s_j(t))) \prod_{k \in \mathcal{N}_{ij}(t)} (1 - p_{ik}(s_k(t))) \quad (11)$$

In our decentralization model, any agent  $j$  is allowed to communicate with its neighboring agents in  $\mathcal{A}_j(t)$ . Therefore, the local information of an agent is the union of the observations of agent  $j$  and the observations of agents  $k \in \mathcal{A}_j(t)$ . In Sec. V, we will provide a precise definition of “information” in terms of observable events such as “agent stops” or “target switches dynamics”.

Returning to problem (9), if its solution is to be obtained through gradient-based methods, then a decentralized approach relies on the ability of an agent  $j$  to determine  $\nabla_j J(\theta)$ , the  $j$ th component of  $\nabla J(\theta)$ , based only on the local information available to it. As we will see next, the fact that the evaluation of  $\nabla_j J(\theta)$  is entirely *event-driven* implies that this issue reduces to the question of *event observability*: if every agent  $j$  can observe all events required to evaluate  $\nabla_j J(\theta)$ , then decentralization is possible.

In order to carry out such an event observability analysis, in the next section we will review the IPA event-driven gradient approach, define all required events, and hence determine explicit conditions under which  $\nabla_j J(\theta)$  can be determined based only on those events local to agent  $j$ .

#### V. INFINITESIMAL PERTURBATION ANALYSIS

As mentioned in Sec. III, the parametric trajectories are selected from a family  $\{s_j(\theta_j, t, s_{j,0}), j = 1, \dots, N\}$  where agents are subject to dynamics (1) and targets to dynamics (4). IPA presented in [19] specifies how changes of the parameter  $\theta$  influence each agent state  $s_j(\theta_j, t, s_{j,0})$  and target state  $R_i(t; \theta)$ , as well as event times  $\tau_k(\theta)$ ,  $k = 1, 2, \dots, K$ , and, ultimately the objective function (9).

It is shown in [19] that in a hybrid system with dynamics  $\dot{x} = f_k(x, \theta, t)$ ,  $k = 0, 1, \dots$ , the trajectory of the state’s partial derivative with respect to  $\theta$  over an interval  $[\tau_k(\theta), \tau_{k+1}(\theta))$  can be written as

$$\frac{d}{dt} x'(t) = \frac{\partial f_k(t)}{\partial x} x'(t) + \frac{\partial f_k(t)}{\partial \theta} \quad (12)$$

for  $t \in [\tau_k, \tau_{k+1})$  with boundary condition:

$$x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)] \tau_k' \quad (13)$$

for  $k = 1, \dots, K$ . We use the Jacobian matrix notation:  $x'(t) \equiv \frac{\partial x(\theta, t)}{\partial \theta}$  and  $\tau_k' \equiv \frac{\partial \tau_k(\theta)}{\partial \theta}$ , for all state and event time

derivatives. For convenience, we set  $\tau_0 = 0$  and  $\tau_{K+1} = T$ . In order to complete the evaluation of  $x'(\tau_k^+)$  in (13), we need to determine  $\tau_k'$ . If the event at  $\tau_k$  is *exogenous* (i.e., independent of  $\theta$ ),  $\tau_k' = 0$ . However, if the event is *endogenous*, there exists a continuously differentiable function  $g_k : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}$  such that  $\tau_k = \min\{t > \tau_{k-1} : g_k(x(\theta, t), \theta) = 0\}$  and, as shown in [19],

$$\tau_k' = -\left[\frac{\partial g_k}{\partial x} f_k(\tau_k^-)\right]^{-1} \left(\frac{\partial g_k}{\partial \theta} + \frac{\partial g_k}{\partial x} x'(\tau_k^-)\right) \quad (14)$$

as long as  $\frac{\partial g_k}{\partial x} f_k(\tau_k^-) \neq 0$ .

We now apply the IPA scheme to our setting in (9), and note that the gradient for each agent  $j$  denoted by  $\nabla_j J(\theta) = \left[\frac{\partial J(\theta, \mathbf{w})}{\partial \theta_j}\right]^\top$  is

$$\nabla_j J(\theta) = \frac{1}{T} \sum_{i=1}^M \sum_{k=0}^K \int_{\tau_k(\theta)}^{\tau_{k+1}(\theta)} \nabla_j R_i(t; \theta) dt \quad (15)$$

**IPA gradient of  $\nabla_j R_i(t; \theta)$ .** We begin by deriving the gradient within any inter-event interval  $[\tau_k, \tau_{k+1})$  over which the dynamics of both agent  $j$  and target  $i$  remain unchanged. The form of this gradient will facilitate the definition of all events at which (13)-(14) will be applied. For notational simplicity, we will henceforth write  $R_i(t)$  instead of  $R_i(t; \theta)$ .

Observing that the first term in (12) vanishes since  $\dot{f}_k(t) = \dot{R}_i(t)$  is not an explicit function of  $R_i(t)$ , we get  $\frac{d}{dt} \frac{\partial R_i(t)}{\partial \theta_j} = \frac{\partial \dot{R}_i(t)}{\partial \theta_j}$ . Then, in view of (4), we have for all  $t \in [\tau_k, \tau_{k+1})$ :

$$\frac{\partial R_i(t)}{\partial \theta_j} = \frac{\partial R_i(\tau_k^+)}{\partial \theta_j} - \begin{cases} 0 & \text{if } R_i(t) = 0, A_i \leq B_i P_i(s(t)) \\ B_i \int_{\tau_k}^t \frac{\partial P_i(s(\tau))}{\partial \theta_j} d\tau & \text{otherwise} \end{cases} \quad (16)$$

The integrand in (16) is obtained from (3):

$$\frac{\partial P_i(s(\tau))}{\partial \theta_j} = \frac{\partial p_{ij}(s_j(\tau))}{\partial \theta_j} \prod_{g \neq j} [1 - p_{ig}(s_g(\tau))] \quad (17)$$

The first term on the right-hand-side of (17) can be obtained from

$$\frac{\partial p_{ij}(s_j(\tau))}{\partial \theta_j} = \begin{cases} \frac{\partial p_{ij}(s_j(\tau))}{\partial d_{ij}} \frac{\partial d_{ij}}{\partial s_j} \frac{\partial s_j}{\partial \theta_j} & \text{if } d_{ij}(t) \leq r_j \\ 0 & \text{if } d_{ij}(t) > r_j \end{cases} \quad (18)$$

where  $d_{ij}(t)$  is the distance between target  $i$  and agent  $j$ :

$$d_{ij}(t) = \sqrt{(s_j^x(t) - x_i^x)^2 + (s_j^y(t) - x_i^y)^2} \quad (19)$$

When  $d_{ij}(t) \leq r_j$ , we have

$$\begin{aligned} \frac{\partial p_{ij}(s_j(\tau))}{\partial \theta_j} &= \frac{\partial p_{ij}(s_j(\tau))}{\partial d_{ij}} \frac{1}{d_{ij}} \left( (s_j^x - x_i^x) \frac{\partial s_j^x}{\partial \theta_j} \right. \\ &\quad \left. + (s_j^y - x_i^y) \frac{\partial s_j^y}{\partial \theta_j} \right) \end{aligned} \quad (20)$$

Note that the term  $\frac{\partial p_{ij}(\tau)}{\partial \theta_j}$  in (17) depends exclusively on information local to agent  $j$  as seen in (18).

As for the product term in (17), it captures the contributions from all agents other than  $j$  in monitoring target  $i$ . Using  $\mathcal{N}_{ij}(t)$  defined in (10), it can be restricted to this set, since for

any agent  $n \notin \mathcal{N}_{ij}(t)$ , we have  $p_{in}(s_n(t)) = 0$ . For notational simplicity, we define this term as

$$G_{ij}(t) \equiv \prod_{n \in \mathcal{N}_{ij}(t)} [1 - p_{in}(s_n(t))] \quad (21)$$

which can be interpreted as a ‘‘collaboration factor’’ involving all agents in  $\mathcal{N}_{ij}(t)$ . Clearly, this is affected by any agent leaving or joining the neighbor set  $\mathcal{N}_{ij}(t)$ , which motivates the definition of an associated event, as detailed below.

Using (21) in (16), we obtain the derivative  $\frac{\partial R_i(t)}{\partial \theta_j}$ ,  $i = 1, \dots, M$ , over any inter-event interval  $[\tau_k, \tau_{k+1})$ :

$$\frac{\partial R_i(t)}{\partial \theta_j} = \frac{\partial R_i(\tau_k^+)}{\partial \theta_j} - \begin{cases} 0 & \text{if } R_i(t) = 0, A_i \leq B_i P_i(s(t)) \\ B_i \int_{\tau_k}^t \frac{\partial p_{ij}(s_j(\tau))}{\partial s_j} \frac{\partial s_j(\tau)}{\partial \theta_j} G_{ij}(\tau) d\tau & \text{otherwise} \end{cases} \quad (22)$$

**Hybrid system event definition.** We are now in a position to define as ‘‘events’’ all switches which can result in changes in the derivatives in (22). These switches may occur as follows: (i) Type I events:  $R_i(t)$  switches from a positive value to 0 or vice versa.

(ii) Type II events: some  $p_{in}(s_n(t))$  switches from a positive value to 0 or vice versa; this is equivalent to a change in sign of  $(d_{in}(t) - r_n)$  in (2) from non-positive to positive or vice versa. Such events can have two effects. First, if  $n \in \mathcal{N}_{ij}(t)$ , it affects the set  $\mathcal{N}_{ij}(t)$  because of the addition or removal of an agent from this neighbor set in (21), hence affecting  $G_{ij}(\tau)$  in (22). Second, if  $n = j$ , recalling (18), this event affects  $\frac{\partial p_{ij}(s_j(\tau))}{\partial d_{ij}}$  through (2), hence also  $\frac{\partial p_{ij}(s_j(\tau))}{\partial \theta_j}$  in (22).

(iii) Type III events: Any agent switches its control and causes a discontinuity in  $u_j(t)$ . Such events affect  $\frac{\partial s_j}{\partial \theta_j}$  in (22).

These three event types are summarized in Table I. Observe that only event types I and III directly affect the dynamics of the corresponding target or agent. Type II events do not change the system dynamics but still may affect the derivative values in (22). In what follows, we consider each of the event types and its effect on the IPA derivatives (22) through (13)-(14).

**Type I events: switches in target state  $R_i(t)$ .** Referring to (4), when  $R_i(t)$  either reaches zero (which can only happen if  $A_i < B_i P_i(s(t))$  based on (4)) or leaves zero, the IPA derivative switches between the two branches in (22). To eliminate the pathological case where  $R_i(t)$  hits zero and leaves immediately, we make the following assumption to ensure that  $R_i(t) = 0$  for a finite amount of time.

**Assumption 1.** If  $R_i(t_0^-) > 0$  and  $R_i(t_0) = 0$  at  $t_0 \in (0, T)$ , there exists an  $\epsilon > 0$  such that  $R_i(t) = 0$  for  $t \in [t_0, t_0 + \epsilon)$ .

TABLE I: Event Definition

| Name         | Description                                | Type |
|--------------|--------------------------------------------|------|
| $\rho_i^0$   | $R_i(t)$ hits 0                            | I    |
| $\rho_i^+$   | $R_i(t)$ leaves 0                          | I    |
| $\pi_{ij}^0$ | $p_{ij}(s_j(t))$ hits 0                    | II   |
| $\pi_{ij}^+$ | $p_{ij}(s_j(t))$ leaves 0                  | II   |
| $\nu_j$      | $u_j(t)$ encounters a discontinuous switch | III  |

Note: events in the table include all  $i = 1, \dots, M$  and  $j = 1, \dots, N$

We denote the event when  $R_i(t)$  reaches zero by  $\rho_i^0$  and the event when it leaves 0 as  $\rho_i^+$  for all  $i = 1, \dots, M$  (see Table I). When such events occur, the dynamics of  $s_j(t)$  in (1) remain unchanged, so it follows from (13) that  $\nabla_j s_j(\tau_k^-) = \nabla_j s_j(\tau_k^+)$ . However, the target dynamics switch between  $\dot{R}_i = A_i - B_i P_i(s(t))$  and  $\dot{R}_i = 0$  and cause discontinuities in  $\nabla_j R_i(t)$  as follows.

**Event  $\rho_i^0$ :** This is an endogenous event because its occurrence depends on the parameter  $\theta$  which dictates switches in  $s(t)$ . We first evaluate  $\tau_k'$  from (14) with  $g_k(R_i(t), t) = R_i(t) = 0$  to get

$$\tau_k' = -\frac{\nabla_j R_i(\tau_k^-)}{A_i - B_i P_i(s(\tau_k^-))} \quad \text{for all } j \quad (23)$$

and then apply (13) to obtain

$$\nabla_j R_i(\tau_k^+) = \nabla_j R_i(\tau_k^-) + [A_i - B_i P_i(s(\tau_k^-)) - 0] \tau_k' \quad (24)$$

Combining (23) and (24), we get

$$\nabla_j R_i(\tau_k^+) = 0 \quad \text{if event } \rho_i^0 \text{ occurs at } \tau_k \quad (25)$$

**Event  $\rho_i^+$ :** This event causes a transition from  $\dot{R}_i(t) = 0$ ,  $t < \tau_k$  to  $\dot{R}_i(t) = A_i - B_i P_i(s(t)) > 0$ ,  $t \geq \tau_k$ . It is easy to see that the dynamics in both (1) and (4) are continuous when this happens and since  $A_i - B_i P_i(s(\tau_k)) = 0$  we have  $\dot{R}_i(\tau_k^-) = \dot{R}_i(\tau_k^+) = 0$ . It follows from (13) that  $\nabla_j R_i(\tau_k^+) = \nabla_j R_i(\tau_k^-)$ . Moreover, since  $R_i(t) = 0$ ,  $\dot{R}_i(t) = 0$ ,  $t < \tau_k$ , we have  $\nabla_j R_i(\tau_k^-) = 0$  and we get

$$\nabla_j R_i(\tau_k^+) = 0 \quad \text{if event } \rho_i^+ \text{ occurs at } \tau_k \quad (26)$$

**Remark 1:** Combining (25) and (26) with (22), we conclude that a  $\rho_i^0$  event occurring at  $t = \tau_k$  resets the value of  $\nabla_j R_i(t)$  to  $\nabla_j R_i(t) = 0$  for all  $j = 1, \dots, N$  regardless of the value  $\nabla_j R_i(\tau_k^-)$  and the state of the agents. Moreover,  $R_i(t) = 0$  and  $\nabla_j R_i(t) = 0$  for  $t > \tau_k$  until the next  $\rho_i^+$  event occurs.

**Type II events: switches in agent sensing function**  $p_{ij}(s_j(t))$ . These events trigger a switch in  $p_{ij}(s_j(t))$  from some positive value to 0 or vice versa. We denote the former event as  $\pi_{ij}^0$  and the latter as  $\pi_{ij}^+$ . The dynamics in both (1) and (4) remain unchanged when this happens (due to the continuity of the sensing function  $p_{ij}(s_j(t))$ ) and it follows from (13) that  $\nabla_j R_i(\tau_k^+) = \nabla_j R_i(\tau_k^-)$  and  $\nabla_j s_j(\tau_k^+) = \nabla_j s_j(\tau_k^-)$ .

**Type III events: switches in agent control**  $u_j(t) \in \{-1, 0, 1\}$ . We denote these events by  $\nu_j$ . Referring to (1), they cause a discontinuous switch in the optimal control values  $\|u_j(t)\| \in \{0, 1\}$  from one to the other. This in turn may cause discontinuities in  $\nabla_j s_j(t)$  at  $t = \tau_k$  which affects (20) and hence (22). The precise expressions for these derivatives depend on the specific parameterization  $\{s_j(\theta_j, t, s_{j,0}), j = 1, \dots, N\}$ . We will provide in Section VII-A these expressions for the 1D case where this parameterization coincides with optimal solutions. For the 2D case, we will provide in Section VII-B the analysis corresponding to the family of elliptical agent trajectories.

**Remark 2:** Observe that  $\nabla_j s_j(t)$  is independent of the states of other agents  $k \neq j$  for which  $\nabla_k s_j(t) = 0$ . This follows from the fact that agents can fully control their movement independent of other agents, and  $\nabla_j s_j(t)$  only depends on parameter and control values known to agent  $j$ .

**Remark 3:** It is clear from the analysis thus far, that the IPA-based gradient is *event-driven*, since all gradient updates occur exclusively at events times  $\tau_k(\theta)$ ,  $k = 1, 2, \dots, K$ . Thus, this approach scales *linearly* with the number of events as opposed to methods like dynamic programming which would scale *exponentially* with the number of agents and targets.

## VI. DECENTRALIZED EVENT-DRIVEN OPTIMIZATION

The set of all events defined in the previous section and summarized in Table I is denoted by  $\mathcal{E}$ . Furthermore, we define the set of all type I and II events as the *target event set*  $\mathcal{E}^T$  (these events pertain to targets) and the set of all type III events as the *agent event set*  $\mathcal{E}^A$  (these events pertain to agents). The subset of  $\mathcal{E}^A$  that contains only events related to agent  $j$  is denoted by  $\mathcal{E}_j^A$ . Similarly, the subset of  $\mathcal{E}^T$  that contains only events related to target  $i$  is denoted by  $\mathcal{E}_i^T$ . We then have:

**Definition 4.** The local event set of any agent  $j$  is the union of agent events  $\mathcal{E}_j^A$  and target events  $\mathcal{E}_i^T$  for all  $i \in \mathcal{T}_j(t)$ :

$$\mathcal{E}_j(t) = \mathcal{E}_j^A \bigcup_{i \in \mathcal{T}_j(t)} \mathcal{E}_i^T \quad (27)$$

Thus,  $\mathcal{E}_j(t)$  consists of events directly observable by agent  $j$ , either because they are entirely under its control or because they are related to targets which are within its sensing range. In contrast, the global event set for agent  $j$  includes all non-neighboring target events in  $\mathcal{E}_i^T$  for all  $i \notin \mathcal{T}_j$  and non-neighboring agent events  $\mathcal{E}_k^A$ , for all  $k \notin \mathcal{A}_j$ . Based on the limited information model of Section IV, we define the *local information set* of agent  $j$ , denoted by  $I_j(t)$ , as follows:

**Definition 5.** The local information set of any agent  $j$  is the union of its local event set and those of its neighbors in  $\mathcal{N}_{ij}(t)$  for all  $i \in \mathcal{T}_j(t)$ :

$$I_j(t) = \mathcal{E}_j(t) \bigcup_{k \in \mathcal{N}_{ij}(t), i \in \mathcal{T}_j(t)} \mathcal{E}_k(t). \quad (28)$$

This set includes all local information necessary for agent  $j$  to evaluate the IPA gradient  $\nabla_j R_i(t)$  for  $i \in \mathcal{T}_j(t)$ . Observe that agent  $j$  does not need to communicate with all its neighbors in  $\mathcal{A}_j(t)$ , but only a subset which includes those neighbors who are sharing the same target(s) as  $j$  at time  $t$ .

Our main decentralization result is presented in Theorem 1 where we show that each agent can evaluate the gradient of the objective function in (9) with respect to its own controllable parameters  $\theta_j$  based on its local information set (28) and only one non-local event. We begin with the following lemma which asserts that the gradient  $\nabla_j R_i(t)$  takes a simple form as long as  $i \notin \mathcal{T}_j(t)$ , i.e., while target  $i$  cannot be sensed by agent  $j$ .

**Lemma 1.** Let  $t \in [t_1, t_2]$  such that  $i \notin \mathcal{T}_j(t)$ . Then,

- 1) If  $R_i(t) > 0$  for all  $t \in [t_1, t_2]$ , then

$$\nabla_j R_i(t) = \nabla_j R_i(t_1^+) \quad (29)$$

- 2) If there exists an event  $\rho_i^0$  at  $\tau \in (t_1, t_2)$ , then

$$\nabla_j R_i(t) = \begin{cases} \nabla_j R_i(t_1^+) & t \in [t_1, \tau) \\ 0 & t \in [\tau, t_2] \end{cases} \quad (30)$$

*Proof:* By the definition of  $\mathcal{T}_j(t)$ , when  $i \notin \mathcal{T}_j(t)$  we have  $\|s_j(t) - x_i\| > r_j$  and  $\frac{\partial p_{ij}(s_j(t))}{\partial s_j} = 0$  for all  $t \in [t_1, t_2]$ . If  $R_i(t) > 0$  for all  $t \in [t_1, t_2]$ , it follows directly from (22) that  $\nabla_j R_i(t) = \nabla_j R_i(t_1^+)$ . Otherwise, there exists an event  $\rho_i^0$  at time  $\tau \in (t_1, t_2)$  which results in  $R_i(\tau) = 0$ . The previous argument applies to  $(t_1, \tau)$  giving  $\nabla_j R_i(t) = \nabla_j R_i(t_1^+)$  for  $t \in [t_1, \tau)$ . According to (25), event  $\rho_i^0$  resets the gradient to  $\nabla_j R_i(\tau) = 0$ . Subsequently, over  $[\tau, t_2]$ , regardless of which of the cases in (22) applies, it holds that  $\nabla_j R_i(t) = 0$ . ■

**Corollary 1.**  $\nabla_j R_i(t)$  is independent of events  $\rho_i^+$  for  $i \notin \mathcal{T}_j(t)$ .

*Proof:* Note that the  $\rho_i^+$  event can only occur after a  $\rho_i^0$  event. The proof is self-evident following Lemma 1. We have  $\nabla_j R_i(t) = 0$  for  $t > \tau$  until target  $i$  joins the target neighborhood of agent  $j$ . Therefore, any non-local  $\rho_i^+$  event that may occur cannot affect  $\nabla_j R_i(t)$ . ■

Lemma 1 and its Corollary imply that agent  $j$  does not need any knowledge of non-neighboring target events except for  $\rho_i^0$  with  $i \notin \mathcal{T}_j(t)$  in order to evaluate its gradient.

We can further establish that the gradient  $\nabla_j J(\theta)$  along the agent trajectory is affected by only local events in  $I_j(t)$  and a small subset of global events.

**Lemma 2.** A sufficient event set to evaluate  $\nabla_j J(\theta)$  is  $I_j(t) \cup \{\rho_i^0 : i \notin \mathcal{T}_j(t)\}$ .

*Proof:* Let  $\tau_k$  be any event time when  $\mathcal{T}_j(\tau_k)$  is altered, i.e., a new target is added to the target neighborhood of agent  $j$  or one is removed from it. From Lemma 1, if  $i \notin \mathcal{T}_j(t)$ , then either  $\nabla_j R_i(t) = \nabla_j R_i(\tau_k)$  and remains constant at this value or  $\nabla_j R_i(t) = 0$ , depending on whether an event  $\rho_i^0$  takes place. It follows from (15) that the objective function gradient can be rewritten as

$$\begin{aligned} \nabla_j J(\theta) &= \frac{1}{T} \sum_{k=0}^K \sum_{i=1}^M \int_{\tau_k}^{\tau_{k+1}} \nabla_j R_i(t) dt \\ &= \frac{1}{T} \sum_{k=0}^K \left( \sum_{i \notin \mathcal{T}_j(\tau_k^+)} \nabla_j R_i(\tau_k) (\tau_{k+1} - \tau_k) + \sum_{i \in \mathcal{T}_j(\tau_k^+)} \int_{\tau_k}^{\tau_{k+1}} \nabla_j R_i(t) dt \right) \end{aligned} \quad (31)$$

The value of  $\nabla_j R_i(\tau_k)$  in the first term of (31) depends on  $\{\rho_i^0 : i \notin \mathcal{T}_j(t)\}$  which is a subset of events non-local to agent  $j$ . The second term of (31) depends only on the local information set events  $I_j(t)$  since target  $i \in \mathcal{T}_j(t)$  is local to agent  $j$ . Therefore,  $I_j(t)$  is a sufficient event set to evaluate  $\nabla_j J(\theta)$ . ■

**Remark 4:** Although an event  $\rho_i^0$  for  $i \notin \mathcal{T}_j(t)$  is non-local to agent  $j$ , it must be observed by at least one agent  $k \neq j$  such that  $i \in \mathcal{T}_k(t)$ . This is because  $\rho_i^0$  can only take place at some time  $\tau_k$  if one or more agents in its neighborhood cause a transition from  $R_i(\tau_k^-) > 0$  to  $R_i(\tau_k) = 0$  in (4). Therefore, such events can be communicated to agent  $j$  through the agent network, possibly with some delay. The implication of Lemma 2 is an “almost decentralized” algorithm in which each agent optimizes its trajectory through the gradient  $\nabla_j J(\theta)$  using only agent local information; the only exception is occasional

target uncertainty depletion events  $\{\rho_i^0 : i \notin \mathcal{T}_j(t)\}$  transmitted to it from other agents.

Returning to the parametric optimization problem (8), a centralized gradient descent algorithm is

$$\theta^{l+1} = \theta^l - \alpha^l \nabla_j J(\theta^l) \quad (32)$$

where  $l = 0, 1, \dots$  is the iteration index and  $\alpha^l$  is a diminishing step-size sequence satisfying  $\sum_{l=0}^{\infty} \alpha^l = \infty, \lim_{l \rightarrow \infty} \alpha^l = 0$ . A decentralized version of (32) by each agent is

$$\theta_j^{l+1} = \theta_j^l - \alpha^l \nabla_j \hat{J}(\theta^l) \quad (33)$$

where  $\nabla_j \hat{J}(\cdot)$  is the estimates gradient of agent  $j$  based on the limited information provided in Lemma 2.

**Theorem 1.** Any centralized solution defined in (8) through (32) can be recovered by (33) in which each agent  $j$  optimizes its trajectory given the following:

- 1) Initial parameters  $\theta_j^0$ ;
- 2) The local information set  $I_j(t)$ ;
- 3) The subset of the global information set  $\{\rho_i^0 : i \notin \mathcal{T}_j(t)\}$ .

*Proof:* The proof is immediate from Lemma 2.  $\nabla_j J(\theta) = \nabla_j \hat{J}(\theta)$  can be shown given conditions 2 and 3. Condition 1 provides initial parameters for each agent trajectory in order to execute (32) and (33). ■

Note that Theorem 1 involves only a small subset of global events. As seen in our simulation results in the next section, ignoring such non-local events will affect the cooperation among agents and increase the final cost. Thus, it can be interpreted as the “price of decentralization” caused by the requirement to limit agent actions to only local information. It is important to point out that the method of Theorem 1 relies on the gradient  $\nabla_j R_i(t)$  for  $i \notin \mathcal{T}_j(t)$  and not on  $R_i(t)$ . In fact, there is no attempt by agent  $j$  to reconstruct or estimate the states of targets  $i \notin \mathcal{T}_j(t)$ ; the only information from such targets is provided through the occasional  $\rho_i^0$  events.

## VII. SIMULATION RESULTS

### A. Decentralized optimization in 1D spaces

Referring to [12], in 1D spaces the optimal control structure is fully characterized by  $u_j^*(t) \in \{1, 0, -1\}$ . It follows that we can parameterize an optimal trajectory so as to determine (i) control switching points in  $[0, L] \subseteq \mathbb{R}$ , where an agent switches its control from  $\pm 1$  to  $\mp 1$  or possibly 0 and (ii) corresponding non-negative dwell times so that the cost in (5) is minimized. In other words, the optimal trajectory of each agent  $j$  is characterized by two parameter vectors: switching points  $\theta_j = [\theta_{j1}, \theta_{j2}, \dots, \theta_{j\Gamma}]$  and dwell times  $\mathbf{w}_j = [w_{j1}, w_{j2}, \dots, w_{j\Gamma'}]$  where  $\Gamma$  and  $\Gamma'$  are finite and can be computed by the given time horizon  $T$ .

**Simulation example.** Three homogeneous agents are allocated to persistently monitor seven targets in a 1D mission space for  $T = 300$  seconds. The targets are located at  $x_i = 5i$  for  $i = 1, \dots, 7$ , as shown by dotted lines in Fig. 3. The uncertainty dynamics in (4) are defined by the parameters  $A_i = 1, B_i = 5$ , with initial values  $R_i(0) = 1$  for  $i = 1, \dots, 7$ . Each agent has a sensing range of  $r = 3$  and is initialized with  $s_j(0) = 0.5(j - 1)$ ,  $u_j(0) = 1$ ,  $\theta_1^0 = [5, 10, 15, 10, 5, \dots]$ ,

$\theta_2^0 = [15, 20, 25, 20, 15, \dots]$ ,  $\theta_3^0 = [25, 30, 35, 30, 25, \dots]$ , and  $w_j^0 = [0.5, 0.5, 0.5, \dots]$  for all  $j = 1, 2, 3$ . The time step in the simulation is 0.05 seconds and the step-size used for the gradient descent in (33) is  $1/n$  where  $n$  is the iteration index. Note that the finite size of this time step ensures that Assumption 1 always holds. Results of the “almost decentralized” method in Theorem 1 are shown in Fig. 3. The plot depicts the optimal trajectories of the three agents determined after 200 iterations of (33). All three agents are moving through periodic cycles dwelling for a short time at each target before moving to the next. The final cost is  $J^* = 37.38$ . The exact same results (not shown here) as in Fig. 3 were also obtained through the centralized scheme (32) where all information is available to every agent. This confirms the effectiveness of the method in Theorem 1.

As pointed out earlier, the method of Theorem 1 does not involve any knowledge by agent  $j$  of the states of targets  $i \notin \mathcal{T}_j(t)$ . This is illustrated in Fig. 4 which shows (in blue) the fraction of time that agent 1 has any information on the state of target 3 because it happens that  $3 \in \mathcal{T}_1(t)$ . The rest of the time (shown in red) agent 1 is unable to accurately estimate the state of this target, but such information is unnecessary. The agent only needs a small subset of its non-local information, as illustrated by the green dots in Fig. 4.

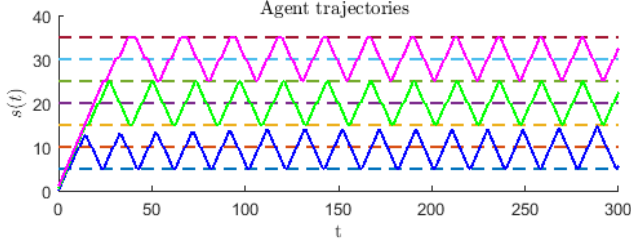


Fig. 3: “Almost decentralized” optimization using Theorem 1. Optimal agent trajectories with final cost  $J^* = 37.38$ .

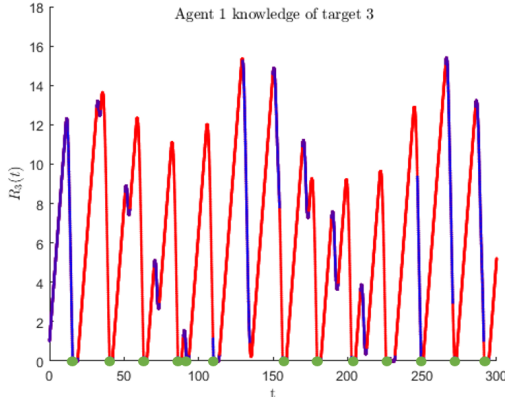


Fig. 4: Red curve:  $R_3(t)$ , the state of target 3. Blue segments:  $R_3(t)$  known to agent 1 when its trajectory includes target 3 in its neighborhood. Green dots: instants when agent 1 receives non-local events  $\rho_3^0$ .

Using the same environment as above and with agents starting with the same initial trajectories, we eliminate the non-local information (condition 3 in Theorem 1) so that each

agent calculates its own IPA-based gradient using only local information in the set  $I_j(t)$ . Figure 5 shows the results after 200 iterations of (33). Note that without non-local information, each agent tends to spend more time dwelling on the local targets instead of better coordinating with the other agents. Therefore, the final cost after convergence increases from 37.38 to 41.66 (thus, the “price of decentralization” here is 4.28). Even though the gradient estimate for agent  $j$  is no longer accurate without the  $\rho_i^0$  event information when  $i \notin \mathcal{T}_j(t)$ , the cost still decreases and converges to a value near the optimal, illustrating the robustness of the IPA-based gradient descent method.

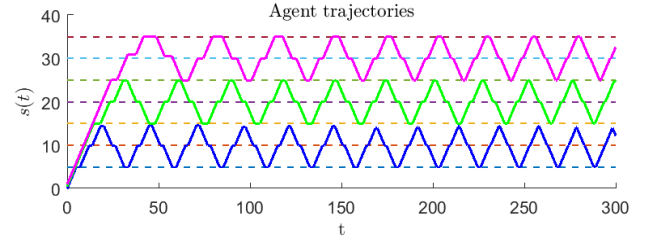


Fig. 5: Fully decentralized optimization without any non-local information. Agent trajectories after decentralized optimization with final cost  $J^* = 41.66$ .

### B. Decentralized optimization in 2D spaces

Motivated by the attractive properties of smooth and periodic trajectories, we choose the family of elliptical agent trajectories where the agent  $j$  trajectory is specified by the parameter set

$$\Theta_j = [X_j, Y_j, a_j, b_j, \varphi_j]$$

such that an agent’s position  $s_j(t) = [s_j^x(t), s_j^y(t)]^\top$  follows the general parametric form of an ellipse:

$$\begin{aligned} s_j^x(t) &= X_j + a_j \cos \rho_j(t) \cos \varphi_j - b_j \sin \rho_j(t) \sin \varphi_j \\ s_j^y(t) &= Y_j + a_j \cos \rho_j(t) \sin \varphi_j + b_j \sin \rho_j(t) \cos \varphi_j \end{aligned} \quad (34)$$

where  $[X_j, Y_j]$  is the center of the ellipse,  $a_j, b_j$  are its major and minor axis respectively,  $\varphi_j \in [0, \pi)$  is the orientation (the angle between the  $x$  axis and the major ellipse axis), and  $\rho_j \in [0, 2\pi)$  is the eccentric anomaly of the ellipse (the phase indicates the position of the agent moving along the ellipse).

The Hamiltonian analysis in Section III shows agents should move with constant maximal speed 1 on the ellipse, hence,  $(\dot{s}_j^x)^2 + (\dot{s}_j^y)^2 = 1$ . The eccentric anomaly then satisfies

$$\dot{\rho}_j(t) = (a_j^2 \sin^2 \rho_j(t) + b_j^2 \cos^2 \rho_j(t))^{-1/2} \quad (35)$$

with a given initial phase position  $\rho_j(0)$ .

The IPA-based gradient follows the analysis in Sec. V, in particular (22) in between events. Given the parametric form of ellipses, the calculation of  $\frac{\partial s_j(t)}{\partial \theta_j}$  is as follows:

$$\begin{aligned} \frac{\partial s_j^x}{\partial X_j} &= 1, & \frac{\partial s_j^y}{\partial X_j} &= 0, & \frac{\partial s_j^x}{\partial Y_j} &= 0, & \frac{\partial s_j^y}{\partial Y_j} &= 1, \\ \frac{\partial s_j^x}{\partial a_j} &= \cos \rho_j(t) \cos \varphi_j, & \frac{\partial s_j^y}{\partial a_j} &= \cos \rho_j(t) \sin \varphi_j, \\ \frac{\partial s_j^x}{\partial b_j} &= -\sin \rho_j(t) \sin \varphi_j, & \frac{\partial s_j^y}{\partial b_j} &= \sin \rho_j(t) \cos \varphi_j, \\ \frac{\partial s_j^x}{\partial \varphi_j} &= -a_j \cos \rho_j(t) \sin \varphi_j - b_j \sin \rho_j(t) \cos \varphi_j, \\ \frac{\partial s_j^y}{\partial \varphi_j} &= a_j \cos \rho_j(t) \cos \varphi_j - b_j \sin \rho_j(t) \sin \varphi_j \end{aligned} \quad (36)$$

and  $\rho_j(t)$  can be calculated through a forward integration of (35). In some cases, we may have  $u_j(t) = 0$  for some  $t$  depending on the parametric description selected (e.g., agent  $j$  stops at specified points on the trajectory for some dwell time to be optimally determined). We have

$$\dot{\rho}_j(t) = \begin{cases} (a_j^2 \sin^2 \rho_j(t) + b_j^2 \cos^2 \rho_j(t))^{-1/2} & \text{if } \|u_j(t)\| = 1 \\ 0 & \text{if } \|u_j(t)\| = 0 \end{cases} \quad (37)$$

**Simulation example.** Two homogeneous agents are allocated to persistently monitor eight targets in a 2D mission space for  $T = 150$  seconds. Targets are located at  $[5, 5]$ ,  $[10, 5]$ ,  $[15, 5]$ ,  $[5, 10]$ ,  $[10, 10]$ ,  $[15, 10]$ ,  $[5, 15]$ ,  $[10, 15]$ ,  $[15, 15]$ . The target uncertainty dynamics in (4) are defined by the parameters  $A_i = 1.5$ ,  $B_i = 10$ , with initial values  $R_i(0) = 1$  for  $i = 1, \dots, 8$ . Each agent has a sensing range of  $r = 5$  and is initialized with parameters  $\theta_1^0 = [5.00, 5.00, 6.50, 5.00, 0, 0]$ ,  $\theta_2^0 = [15.00, 15.00, 6.50, 5.00, 1.57, 0]$ . The time step is 0.05 seconds and the step-size used for the gradient descent in (33) is  $1/n$  where  $n$  is the iteration index. For simplicity, we set all dwell times to zero for all agents. The “almost decentralized” method in Theorem 1 gives the results shown in Fig. 6. The top plots show the initial trajectories of the two agents and the optimal trajectories determined after 300 iterations of (33). The bottom plot shows the overall cost  $J(\theta)$  as a function of iteration number. The final cost is  $J^* = 85.59$ . The exact same results (not shown here) as in Fig. 6 were also obtained through the centralized scheme (32) where all information is available to both agents.

Next we eliminate the non-local information (condition 3 in Theorem 1). Figure 8 shows the results after 300 iterations of (33). Without non-local information, agents tend to cover more targets instead of better coordinating with the other agents. Therefore, the final cost after convergence increases from 85.59 to 92.56 (the “price of decentralization” here is 6.97). Even though the gradient estimate for agent  $j$  is no longer as accurate as the centralized case without the  $\rho_i^0$  events when  $i \notin \mathcal{T}_j(t)$ , the cost decreases and once again converges to a local optimal.

The running times of the almost-decentralized result are 0.4 seconds per iteration for both the 1D and the 2D cases on a MacBook Pro 2016 2.9 GHz Dual-Core Intel Core i5 using

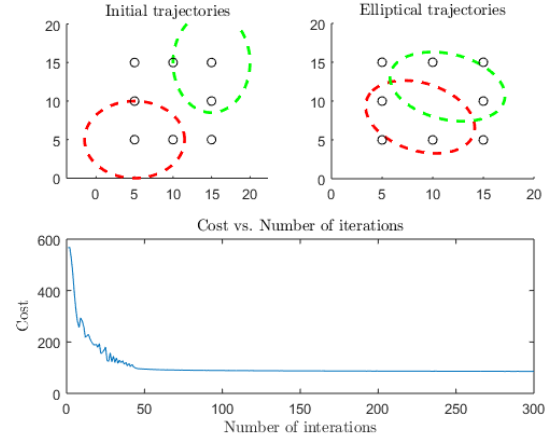


Fig. 6: “Almost decentralized” optimization using Theorem 1. Top left plot: initial agent trajectories: agent 1 in red and agent 2 in green. Top right plot: optimal elliptical trajectories. Bottom plot: cost as a function of number of iterations with  $J^* = 85.59$ .

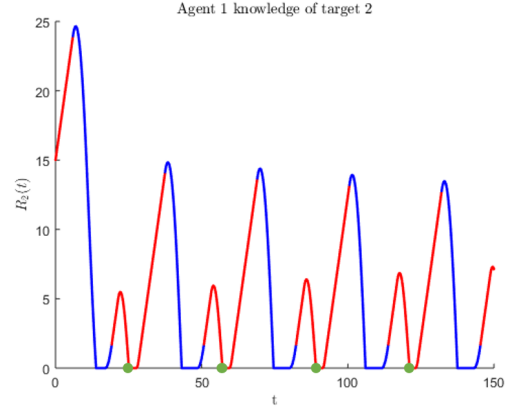


Fig. 7: The state of target  $R_2(t)$  at  $(10, 5)$ . Blue segments:  $R_2(t)$  known to agent 1 when target 2 is in its neighborhood. Green dots: instants when agent 1 receives non-local events  $\rho_2^0$ .

MATLAB. The computational complexity does not increase dramatically from 1D to 2D as the underlying IPA-based gradient calculation scales in the number of events and not the state space. Comparing the results in Fig. 6 and Fig. 8, we can see that the (non-local) target depletion events  $\rho_i^0$  for  $i = 1, \dots, M$  have a more significant effect towards agent performance compared to 1D cases. Moreover, the result obtained using Theorem 1 does not involve any knowledge by agent  $j$  of the states of targets  $i \notin \mathcal{T}_j(t)$  as shown in Fig. 7. The blue segments show the time that agent 1 has information on the state of target 2 because it happens that  $2 \in \mathcal{T}_1(t)$ . The rest of the time (shown in red) agent 1 is unable to accurately estimate the state of this target, but such information is unnecessary. As illustrated by the green dots, the agent only needs a small subset of its non-local information.

Finally, laboratory-based implementations with proof-of-concept experiments of our persistent monitoring algorithms can be found in [22], [23].

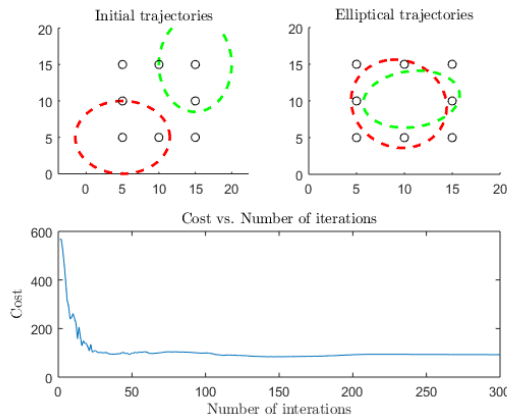


Fig. 8: Fully decentralized optimization without any non-local information. Top left plot: initial agent trajectories. Top right plot: optimal elliptical trajectories. Bottom plot: cost as a function of number of iterations with  $J^* = 92.56$ .

### VIII. CONCLUSIONS AND FUTURE WORK

We have shown that an optimal centralized solution of the persistent monitoring problem can be recovered by an event-driven “almost decentralized” algorithm which significantly reduces communication costs while yielding the same performance. In particular, we have shown that the ability to decentralize the solution is reduced to one of event observability, i.e., whether an agent can observe all events it requires to evaluate its local objective function event-driven IPA gradient. In addition, this analysis allows us to quantify the “price of decentralization” by explicitly measuring the loss in performance when some of the events required for this evaluation are not locally available. Our analysis of IPA gradient applies to any 2D-trajectories as long as these trajectories have a parametric form and the number of parameters is finite. Ongoing research is to adopt this decentralized parameterization framework to different objective functions (e.g., the covariance of target values) and to improve the algorithm developed by handling communication delays.

### REFERENCES

- [1] N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang, “Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in monterey bay,” *Journal of Field Robotics*, vol. 27, no. 6, pp. 718–740, 2010.
- [2] H. Choset, “Coverage for robotics—a survey of recent results,” *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.
- [3] N. Michael, E. Stump, and K. Mohta, “Persistent surveillance with a team of mavs,” in *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots Systems*, 2011, pp. 2708–2714.
- [4] S. L. Smith, M. Schwager, and D. Rus, “Persistent monitoring of changing environments using a robot with limited range sensing,” in *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2011, pp. 5448–5455.
- [5] J. Yu, S. Karaman, and D. Rus, “Persistent monitoring of events with stochastic arrivals at multiple stations,” *IEEE Trans. on Robotics*, vol. 31, no. 3, pp. 521–535, 2015.
- [6] S. K. K. Hari, S. Rathinam, S. Darbha, K. Kalyanam, S. G. Manyam, and D. Casbeer, “The generalized persistent monitoring problem,” in *Proc. of American Control Conference*, 2019, pp. 2783–2788.

- [7] V. A. Huynh, J. J. Enright, and E. Frazzoli, “Persistent patrol with limited-range on-board sensors,” in *Proc. IEEE Conf. on Decision and Control*, 2010, pp. 7661–7668.
- [8] S. L. Smith, M. Schwager, and D. Rus, “Persistent Robotic Tasks: Monitoring and Sweeping in Changing Environments,” *IEEE Trans. on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.
- [9] X. Lin and C. G. Cassandras, “An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces,” *IEEE Trans. on Autom. Contr.*, vol. 60, no. 6, pp. 1659–1664, 2015.
- [10] E. Stump and N. Michael, “Multi-robot persistent surveillance planning as a vehicle routing problem,” in *Proc. IEEE Conf. on Automation Science and Engineering*, 2011, pp. 569–575.
- [11] N. Rezazadeh and S. S. Kia, “A sub-modular receding horizon approach to persistent monitoring for a group of mobile agents over an urban area,” *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 217–222, 2019.
- [12] N. Zhou, X. Yu, S. B. Andersson, and C. G. Cassandras, “Optimal event-driven multiagent persistent monitoring of a finite set of data sources,” *IEEE Trans. on Autom. Control*, vol. 63, no. 12, pp. 4204–4217, 2018.
- [13] X. Lan and M. Schwager, “Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields,” in *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2013, pp. 2415–2420.
- [14] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, “Provably correct persistent surveillance for unmanned aerial vehicles subject to charging constraints,” in *Experimental Robotics*. Springer, 2016, pp. 605–619.
- [15] C. Song, L. Liu, G. Feng, and S. Xu, “Optimal control for multi-agent persistent monitoring,” *Automatica*, vol. 50, no. 6, pp. 1663–1668, 2014.
- [16] M. Zhong and C. G. Cassandras, “Asynchronous distributed optimization with event-driven communication,” *IEEE Trans. on Autom. Control*, vol. 55, no. 12, pp. 2735–2750, 2010.
- [17] W. Ren and N. Sorensen, “Distributed coordination architecture for multi-robot formation control,” *Robotics and Autonomous Systems*, vol. 56, no. 4, pp. 324–333, 2008.
- [18] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.
- [19] C. G. Cassandras, Y. Wardi, C. G. Panayiotou, and C. Yao, “Perturbation analysis and optimization of stochastic hybrid systems,” *European Journal of Control*, vol. 16, no. 6, pp. 642–661, 2010.
- [20] Y. Wardi, R. Adams, and B. Melamed, “A unified approach to infinitesimal perturbation analysis in stochastic flow models: the single-stage case,” *IEEE Trans. on Autom. Contr.*, vol. 55, no. 1, pp. 89–103, 2010.
- [21] N. Zhou, C. G. Cassandras, X. Yu, and S. B. Andersson, “Decentralized event-driven algorithms for multi-agent persistent monitoring tasks,” in *Proc. IEEE Conference on Decision and Control*, 2017, pp. 4064–4069.
- [22] “<https://www.bu.edu/codes/research/1246-2/persistent-monitoring/>.”
- [23] “<https://www.bu.edu/codes/2019/05/07/smart-transportation-in-future-cities/>.”



**Nan Zhou** received the B.S. degree in control engineering from Zhejiang University, Hangzhou, China, in 2011, the M.S. degree in control engineering from University of Chinese Academy of Sciences, Beijing, China, in 2014, and the Ph.D. degree in systems engineering from Boston University, Boston, MA, USA, in 2019. He is currently a senior software engineer at MathWorks Inc., Natick, MA, USA. His research interests include control and optimization of multi-agent systems with applications in robotics and smart transportation.



**Christos G. Cassandras** (F'96) received the B.S. degree in engineering and applied science from Yale University, New Haven, CT, USA, in 1977, the M.S.E.E. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1978, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, USA, in 1979 and 1982, respectively. He was with ITP Boston, Inc., Cambridge, from 1982 to 1984, where he was involved in the design of automated manufacturing systems. From 1984 to 1996, he was a

faculty member with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA, USA. He is currently a Distinguished Professor of Engineering with Boston University, Brookline, MA, USA, the Head of the Division of Systems Engineering, and a Professor of Electrical and Computer Engineering. He serves on several editorial boards and has been a Guest Editor for various journals. He specializes in the areas of discrete event and hybrid systems, cooperative control, stochastic optimization, and computer simulation, with applications to computer and sensor networks, manufacturing systems, and transportation systems. He has authored over 450 refereed papers in these areas, and six books. Dr. Cassandras is a Fellow of the International Federation of Automatic Control (IFAC). He was a recipient of several awards, including the 2011 IEEE Control Systems Technology Award, the 2006 Distinguished Member Award of the IEEE Control Systems Society, the 1999 Harold Chestnut Prize (IFAC Best Control Engineering Textbook), a 2011 prize, and a 2014 prize for the IBM/IEEE Smarter Planet Challenge competition, the 2014 Engineering Distinguished Scholar Award at Boston University, several honorary professorships, a 1991 Lilly Fellowship, and a 2012 Kern Fellowship. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL from 1998 to 2009. He was the President of the IEEE Control Systems Society in 2012. He is also a member of Phi Beta Kappa and Tau Beta Pi.



**Xi Yu** received the B.S. degree and the Dipl.-Ing. degree both in Mechanical Engineering from Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2010 and 2011, and the Ph.D. degree in Mechanical Engineering from Boston University, Boston, MA, USA, in 2018. She is currently a post-doc researcher with the department of Mechanical Engineering and Applied Mechanics at University of Pennsylvania, Philadelphia, PA, USA. Her research interests include control and coordination of multi-robot systems, especially for long term and large

scale monitoring tasks. She is a member of the IEEE Control Systems Society.



**Sean B. Andersson** received the B.S. degree in engineering and applied physics from Cornell University, Ithaca, NY, USA, in 1994, the M.S. degree in mechanical engineering from Stanford University, Stanford, CA, USA, in 1995, and the Ph.D. degree in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 2003. He has worked at AlliedSignal Aerospace and Aerovironment, Inc. and is currently a Professor of mechanical engineering and of systems engineering with Boston University, Boston, MA, USA. His

research interests include systems and control theory with applications in scanning probe microscopy, dynamics in molecular systems, and robotics. He received an NSF CAREER award in 2009, is a senior member of the IEEE, and was an associate editor for the IEEE Transactions on Automatic Control (2014-2018) and for the SIAM Journal on Control and Optimization (2013-2018).