Optimal Threshold-Based Distributed Control Policies for Persistent Monitoring on Graphs

Nan Zhou[1], Christos G. Cassandras[1,2], Xi Yu[3], and Sean B. Andersson[1,3]

***Abstract*— We consider the optimal multi-agent persistent monitoring problem defined by a team of cooperating agents visiting a set of nodes (targets) on a graph with the objective of minimizing a measure of overall node state uncertainty. The solution to this problem involves agent trajectories defined both by the sequence of nodes to be visited by each agent and the amount of time spent at each node. We propose a class of distributed threshold-based parametric controllers through which agent transitions from one node to the next are controlled by thresholds on the node uncertainty. The resulting behavior of the agent-target system is described by a hybrid dynamic system. This enables the use of Infinitesimal Perturbation Analysis (IPA) to determine on-line optimal threshold parameters through gradient descent and thus obtain optimal controllers within this family of threshold-based policies. Simulations are included to illustrate our results and compare them to optimal solutions derived through dynamic programming.**

I. Introduction

The cooperative multi-agent *persistent monitoring* problem arises when agents are tasked to monitor a dynamical environment which cannot be fully covered by stationary agents. Thus, persistent monitoring differs from traditional consensus [1] and coverage [2] problems due to the continuous need to explore changes in the environment. In many cases, this exploration process leads to the discovery of various "points of interest", which, once detected, become "targets" that need to be perpetually monitored. This paradigm applies to surveillance [3], sampling and environmental monitoring [4]. It also finds use in particle tracking in molecular biology to understand the dynamics and interactions of macromolecules [5], [6]. In contrast to sweep coverage and patrolling [7], the problem we address here focuses on a *finite number* of targets. The goal of agents is to collect information from each target so as to reduce a metric of uncertainty about its state. This uncertainty increases while no agent is present in its vicinity and decreases when it is being "sensed" by one or more agents. Thus, the objective is to minimize an overall measure of target uncertainty by controlling the movement of all agents.

Our previous work [8] considered this problem in 1D and showed that the solution can be reduced to a parametric controller form. In particular, the optimal agent trajectories are characterized by a finite number of points where each agent switches direction and by a dwell time at each such point. However, in 2D, such parametric representations for *optimal* agent trajectories no longer hold [9]. Nonetheless, various forms of parametric trajectories (e.g., ellipses, Lissajous curves, interconnected linear segments) offer an alternative [9], [10]. These approaches limit agent trajectories to certain forms which, while they possess desirable properties (e.g., periodicity), cannot always capture the dynamic changes in target uncertainties and may lead to poor local optima.

In this paper, we adopt a higher-level point of view whereby targets are nodes in a graph and their connectivity defines feasible paths in the graph. This graph-based model accounts for physical constraints such as obstacles in the 2D space which the graph is designed to avoid. It is also suitable for problems where agent motion is already constrained by a graph, as in a ground transportation network. Moreover, it allows us to deal with planning problems (e.g. air traffic scheduling) without tracking the detailed agent motion at a higher resolution. An agent trajectory in such a graph is specified by a sequence of nodes and an associated dwell time at each node in the sequence. The controller associated with each agent determines (i) the dwell time at the current node and (ii) the next node to be visited. We propose a class of event-driven controllers, which is distributed and scalable, based on a set of *threshold parameters* associated with the target uncertainties. When compared to the actual value of the target, thresholds are controlled to provide information about the importance of visiting a specific target. By adjusting thresholds, we control agent behavior in terms of target visiting and dwelling and, therefore, optimize agent performance within the parametric family considered.

The contribution of this paper lies in the graph-based setup of the 2D persistent monitoring problem, the formulation of a threshold-based parametric optimization problem, and a solution approach based on Infinitesimal Perturbation Analysis (IPA) [11] and gradient descent. Our approach is *distributed* since the decisions made by an agent at some node are based on uncertainty states of neighboring nodes only. Moreover, we exploit the event-driven nature of IPA to also render it *scalable* in the number of *events* in the system and not the size of the state space (in contrast to solutions dependent on dynamic programming). An additional contribution is to show that in the case of a single-agent system the IPA gradient is monotonic in the thresholds involved which implies a simple optimal structure: the agent visiting a node should reduce the uncertainty state to zero before moving to the next node. This is consistent with a similar earlier result established in [12].

The authors are affiliated with [1]Division of Systems Engineering, [2]Department of Electrical and Computer Engineering, [3]Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA. E-mail: {nanzhou,cgc, xyu,sanderss}@bu.edu.

* The work of Cassandras and Zhou is supported in part by NSF under grants ECCS-1509084, CNS-1645681, and IIP-1430145, by AFOSR under grant FA9550-15-1-0471, by ARPA-Es NEXTCAR program under grant DE-AR0000796, by MathWorks and by Bosch. The work of Andersson and Yu is supported in part by NSF through grants ECCS-1509084 and CMMI-1562031.

II. Problem formulation

Consider N agents and M targets in a graph $G = (V, E)$ where the set of vertices V (nodes) is defined by an indexed list of targets $V = \{1, \ldots, M\}$ and the set of edges (links) E contains all feasible direct connections between them. The neighborhood of a node i is defined to be the set $\mathcal{N}_G(i) = \{j : e_{ij} \in E\}$. The matrix $\Delta = [\delta_{ij}]$ defines the travel time over every edge, e.g., δ_{ij} is the travel time over edge e_{ij}.

Target uncertainty model. We define uncertainty functions $R_i(t)$ associated with targets (nodes) $i = 1, \ldots, M$, with the following properties: (i) $R_i(t)$ increases with a prespecified rate A_i if no agent is visiting it, (ii) $R_i(t)$ decreases with a rate $B_i N_i(t)$ where B_i is the rate at which an agent collects data from target i, hence decreasing its uncertainty state, and $N_i(t)$ is the number of agents dwelling at target i at time t, and (iii) $R_i(t) \geq 0$ for all t. We model the target uncertainty state dynamics as follows:

$$\dot{R}_i(t) = \begin{cases} 0 & \text{if } R_i(t) = 0 \text{ and } A_i \leq B_i N_i(t) \\ A_i - B_i N_i(t) & \text{otherwise} \end{cases} \quad (1)$$

Agent model. In the graph topology, an agent is either at a node or at an edge. We define the state of agent a, for $a = 1, \ldots, N$, to be $\mathbf{x}_a(t) = (s_a(t), v_a(t))$ where $s_a(t) \in V$ is a member of the node set V and $v_a(t) \in \{0, 1\}$ indicates whether the agent is dwelling at the node or traveling to it (e.g., $(i, 0)$ means the agent is physically located at a vertex i and $(i, 1)$ means the agent is traveling to node i from some previous node). The travel time matrix Δ gives the transition time over each edge. The agent controller's role when visiting some node is to determine the dwelling time and the index of next destination, denoted by $u_a(t)$. Figure 1 shows a typical control trajectory and helps pinpoint the behavior of each agent controller. The trajectory consists of a sequence of intervals $[t_{a,k}, t_{a,k+1})$ where the agent's node visits are indexed by $k = 1, 2, \ldots$ and $t_{a,k}$ is the time of the k-th visit at any node. This interval contains the agent's dwelling time $d_{a,k}$ and the travel time $\delta_{a,k}$ to the next node. Note that on any trajectory:

$$t_{a,k+1} = t_{a,k} + d_{a,k} + \delta_{a,k}$$

and $\delta_{a,k} = \delta_{ij}$, the ij-entry in Δ, where $i = s_a(t_{a,k} + d_{a,k})$ is the current node and $j = u_a(t_{a,k} + d_{a,k})$ is the destination. $u_a(t)$ is a piecewise constant right-continuous function of time. According to the graph topology, the control $u_a(t)$ is selected from the feasible control set $\mathcal{U}_a(t) = \{\mathcal{U}_{a,1}(t), \ldots, \mathcal{U}_{a,M}(t)\}$ such that at any node i:

$$\mathcal{U}_{a,i}(t) = \begin{cases} \{i\} \cup \mathcal{N}_G(i) & \text{if } \mathbf{x}_a(t) = (i, 0) \\ \{i\} & \text{if } \mathbf{x}_a(t) = (i, 1) \end{cases} \quad (2)$$

where $\mathcal{N}_G(i)$ is the neighborhood of node i which contains all the directly connected nodes to node i.

Objective function. Our goal is to determine the optimal control $u_a^*(t)$ for all agents under which the average uncertainty metric in (3) across all targets is minimized over T. We formulate the following optimal control problem:

$$\mathbf{P1}: \min_{\substack{u_a(t) \in \mathcal{U}_a(t) \\ \text{for } a=1,\ldots,N}} J = \frac{1}{T} \int_0^T \sum_{i=1}^{M} R_i(t)\, dt \quad (3)$$

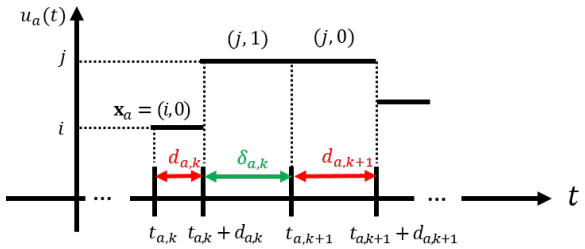


Fig. 1: An agent control trajectory: $d_{a,k}$ is the k-th dwell time and $\delta_{a,k}$ is the k-th travel time. Node i and j are the agent's k-th and $k+1$-th visiting nodes.

subject to the target dynamics (1) and the agent controls (2).

Observe that the control $u_a(t)$ switches only at times $t_{a,k} + d_{a,k}$ for $k = 1, 2....$ The condition under which such a switch occurs may generally be expressed by a switching function $g_{i,j}(\mathbf{x}(t), \mathbf{R}(t)) \leq 0$ associated with a transition from node i to node j, which is a function of the overall state consisting of all agents $\mathbf{x}(t) = [\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)]$ and all targets $\mathbf{R}(t) = [R_1(t), \ldots, R_M(t)]$. Let us define a guard function $\tau_{a,k}^j = \inf_{t \geq t_{a,k}} \{g_{i,j}(\mathbf{x}(t), \mathbf{R}(t)) = 0\}$ and set

$$t_{a,k} + d_{a,k} = \min_{j \in \mathcal{N}_G(i)} \{\tau_{a,k}^j\}$$

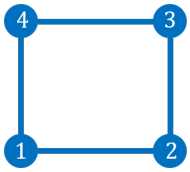
so that the change in the agent's node assignment occurs at the earliest time that one of the switching functions satisfies $g_{i,j}(\mathbf{x}(t), \mathbf{R}(t)) = 0$. Thus, the task of the controller is to determine optimal switching functions for all $j \in \mathcal{N}_G(i)$ whenever $\mathbf{x}_a(t) = (i, 0)$ and then evaluate $\min_{j \in \mathcal{N}_G(i)} \{\tau_{a,k}^j\}$ to specify the optimal dwelling time $d_{a,k}^*$. Therefore,

$$u_a^*(t) = \begin{cases} i & t \in [t_{a,k}^*, t_{a,k}^* + d_{a,k}^*) \\ \arg \min_{j \in \mathcal{N}_G(i)} \{\tau_{a,k}^j\} & t \in [t_{a,k}^* + d_{a,k}^*, t_{a,k+1}^*) \end{cases} \quad (4)$$

The complete state of this system is defined by $\mathbf{x}(t)$ and $\mathbf{R}(t)$ so that the control can be expressed as $u_a(\mathbf{x}(t), \mathbf{R}(t))$. In effect, whenever $\mathbf{x}_a(t) = (i, 0)$, the state space defined by all feasible values of $[\mathbf{x}(t), \mathbf{R}(t)]$ is partitioned into $|\mathcal{N}_G(i)| + 1$ regions, denoted by $\mathcal{R}_i$ and $\mathcal{R}_j$, $j \in \mathcal{N}_G(i)$. The controller keeps the agent at node i as long as $[\mathbf{x}(t), \mathbf{R}(t)] \in \mathcal{R}_i$ and switches to $u_a(t) = j \in \mathcal{N}_G(i)$ as soon as the state vector transitions to a new region $\mathcal{R}_j$. Thus, the optimization problem consists of determining an optimal partition for all $i = 1, \ldots, M$ through $g_{i,j}^*(\mathbf{x}(t), \mathbf{R}(t))$ for all $j \in \mathcal{N}_G(i)$.

Parametric control. Designing an optimal feedback controller for **P1** is generally intractable where optimal partitions of the state space must be determined whenever an agent visits a node. An alternative is to seek a parameterization of these partitions through parameters $\mathbf{\Theta}$ so as to ultimately replace **P1** by a problem requiring the determination of a finite set of optimal parameters such that $\mathbf{\Theta}^* = \arg\min J(\mathbf{\Theta})$.

We introduce *threshold parameters* associated with a node i which, when compared to the actual value of $R_i(t)$, provide information about the importance of visiting this node next so that an agent can evaluate the control in (4). We set thresholds for each agent at each node, thus rendering our control policy rich enough to include both node states and



$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \infty & \theta_{14} \\ \theta_{21} & \theta_{22} & \theta_{23} & \infty \\ \infty & \theta_{3,2} & \theta_{3,3} & \theta_{3,4} \\ \theta_{41} & \infty & \theta_{43} & \theta_{44} \end{bmatrix}$$

Fig. 2: A 1-agent 4-target example. The node topology graph is shown on the left and the threshold matrix is on the right. A threshold parameter is set to ∞ when there is no direct path between the corresponding nodes. An agent located at node 1 uses a state space partition parametrized by θ_{11}, θ_{12} and θ_{14}. The agent index (superscript) is omitted for this single agent case.

agent states. We organize the node thresholds associated with agent a into an $M \times M$ matrix $\mathbf{\Theta}^a$ where a row represents the current node visited by the agent and a column represents a potential next node to visit (see Fig. 2). We then define the specific threshold-based controller family considered. The starting point is to define a state space region forcing the agent to remain at node i. This is expressed through the condition $R_i(t) > \theta^a_{ii}$. When this is no longer met, i.e., the uncertainty state at node i is sufficiently low with respect to θ^a_{ii}, the agent may be assigned to a new node $j \neq i$ as long as its uncertainty state exceeds another threshold, i.e., $R_j(t) \geq \theta^a_{ij}$. Since there may be several nodes in the neighborhood of i whose uncertainty states are high relative to their associated thresholds, we prioritize nodes in the neighborhood of i by defining an ordered set for agent a as follows:

$$\mathcal{N}^a_G(i) = \{j_k \in \mathcal{N}_i : j_1, \ldots, j_k, \ldots, j_{D_i}\} \tag{5}$$

where D_i is the degree of vertex i. The prioritization used may depend on several factors (e.g., importance of a target, node proximity, travel distance, etc). We now define the threshold-based control to specify $u_a(t; \mathbf{\Theta})$ in (4) as follows:

$$u_a(t;\mathbf{\Theta}) = \begin{cases} i & \text{if } R_i(t) > \theta^a_{ii} \text{ or} \\ & R_j(t) < \theta^a_{ij} \text{ for all } j \in \mathcal{N}^a_G(i) \\ \arg\min\limits_{k \atop \text{s.t.} j_k \in \mathcal{N}^a_G(i)} & R_{j_k} \geq \theta^a_{ij_k} \qquad \text{otherwise} \end{cases} \tag{6}$$

Under (6), the agent first decreases $R_i(t)$ through (1) below the threshold θ^a_{ii} before moving to another node in the neighbor set $\mathcal{N}^a_G(i)$ with the minimum index k whose associated state uncertainty value exceeds the threshold $\theta^a_{ij_k}$. If no such neighbor exists, the agent remains at the current node maintaining its uncertainty state under the given threshold level. All agent behaviors are therefore entirely governed by $\mathbf{\Theta}$ through (6), which also implicitly determines the dwell time of the agent at each node.

Remark 1. The controller in (6) is designed to be *distributed* by considering only the states of neighboring nodes and not those of other nodes or of other agents. As such, it is limited to a one-step look-ahead policy. However, it can be extended to a richer family of multi-step look-ahead policies.

Through (6), **P1** is reduced to a parametric optimization problem of determining the optimal thresholds in $\mathbf{\Theta}^\star$ under which the cost function in (3) is minimized. Moreover, the resulting agent and node behavior defines a hybrid system: the node dynamics in (1) switch between the mode where $\dot{R}_i(t) = 0$ and $\dot{R}_i(t) = A_i - B_i N_i(t)$, while the agent control in (4) switches whenever there is a sign change in some expression of the form $(R_j(t) - \theta^a_{ij})$ as seen in (6). We use τ_k to denote the time instant when any of the state variables experiences a mode switch (these will be explicitly defined as "events" later) and $\tau_0 = 0, \tau_K = T$ to denote the beginning and the end of the time horizon. After rewriting the cost in (3) as the sum of costs over all intervals $[\tau_k, \tau_{k+1})$ for $k = 0, \ldots, K-1$ as shown in (7), we have transformed the optimal control problem **P1** into a simpler parametric optimization problem **P2** as follows:

$$\mathbf{P2}: \min_{\mathbf{\Theta} \geq \mathbf{0}} \quad J(\mathbf{\Theta}) = \frac{1}{T} \sum_{i=1}^{M} \sum_{k=0}^{K-1} \int_{\tau_k(\mathbf{\Theta})}^{\tau_{k+1}(\mathbf{\Theta})} R_i(t)\, dt \tag{7}$$

subject to target uncertainty dynamics (1), agent controls (6).

Remark 2. Using the optimal thresholds $\mathbf{\Theta}^\star$, the optimal dwell times and target visiting sequences can both be determined online while executing the control policy (6).

III. Infinitesimal Perturbation Analysis (IPA)

Infinitesimal Perturbation Analysis (IPA) specifies how changes in the parameter $\mathbf{\Theta}$ influence event times $\tau_k(\mathbf{\Theta})$, $k = 1, 2, \ldots$, the agent trajectories $\mathbf{x}(\mathbf{\Theta}, \mathbf{x}_0, \mathbf{R}_0)$, and ultimately the cost (7). Let $\{\tau_k(\theta)\}$, $k = 0, \ldots, K-1$, denote the occurrence times of all events in the state trajectory of a hybrid system with dynamics $\dot{x} = f_k(x, \theta, t)$ over an interval $[\tau_k(\theta), \tau_{k+1}(\theta))$, where $\theta \in \Theta$ is some parameter vector and Θ is a given compact, convex set. We set $\tau_0 = 0$, $\tau_K = T$, and 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. It is shown in [11] that

$$\frac{d}{dt} x'(t) = \frac{\partial f_k(t)}{\partial x} x'(t) + \frac{\partial f_k(t)}{\partial \theta}, \tag{8}$$

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

$$x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)]\tau'_k \tag{9}$$

for $k = 1, ..., K$. In order to complete the evaluation of $x'(\tau_k^+)$ in (9), we need to determine τ'_k. If the event at τ_k is *exogenous* (i.e., independent of θ), $\tau'_k = 0$. However, if the event is *endogenous*, there exists a continuously differentiable guard function $g_k : \mathbb{R}^n \times \Theta \to \mathbb{R}$ such that $\tau_k = \min\{t > \tau_{k-1} : g_k(x(\theta, t), \theta) = 0\}$ and

$$\tau'_k = -[\frac{\partial g_k}{\partial x} f_k(\tau_k^-)]^{-1} (\frac{\partial g_k}{\partial \theta} + \frac{\partial g_k}{\partial x} x'(\tau_k^-)) \tag{10}$$

as long as $\frac{\partial g_k}{\partial x} f_k(\tau_k^-) \neq 0$ (details can be found in [11]).

Differentiating the cost $J(\mathbf{\Theta})$ in **P2**, we obtain

$$\nabla J(\mathbf{\Theta}) = \frac{1}{T} \sum_{i=1}^{M} \sum_{k=0}^{K} \int_{\tau_k(\mathbf{\Theta})}^{\tau_{k+1}(\mathbf{\Theta})} \nabla R_i(t)\, dt \tag{11}$$

where ∇ is the gradient operator. We first derive the integrand $\nabla R_i(t)$ in (11) for all i and then integrate over $[0, T]$ to obtain $\nabla J(\mathbf{\Theta})$. The following lemma shows that

the integrand $\nabla R_i(t)$ remains constant between any two consecutive events and can be updated only at event time. This establishes the fully *event-driven* nature of our IPA-based gradient algorithm. Due to space limitations all proofs are omitted but can be found in [13].

Lemma 1. *$\nabla R_i(t)$ remains constant for $t \in [\tau_k, \tau_{k+1}), k = 0, \dots K-1$.*

In the following, we will specify the discontinuities of $\nabla R_i(t)$ at each event time τ_k. We first define four types of "target events" (labeled Event 1 to 4) in this hybrid system corresponding to $R_i(t)$ crossing some threshold value from above/below, or reaching the value $R_i(t) = 0$ from above or leaving the value $R_i(t) = 0$. Since each agent's movement is governed by the target thresholds, a target event may induce an agent departure, denoted by **DEP**, occurring at $t_{a,k} + d_{a,k}$ in (6). In turn, this event will induce this agent's arrival event at the next node visited, denoted by **ARR**. The process of how events can induce other events is detailed next and is graphically summarized in Fig. 3.

For notational simplicity, we use $\downarrow=$ as an operator indicating that the value on its left-hand-side reaches the value on its right-hand-side from above. Similarly, $\uparrow=$ means reaching from below, and $=\uparrow$ means increasing from the value on the right-hand-side. In addition, since the derivative with respect to $\boldsymbol{\Theta}$ is updated differently at different entries, we use p, q, z to indicate the pq-entry of $\boldsymbol{\Theta}^z$, and the operator $(\cdot)^z_{pq}$ to indicate the pq-entry of a matrix indexed by superscript z, i.e., $(\tau'_k)^z_{pq} = \frac{\partial \tau_k}{\partial \boldsymbol{\Theta}^z_{pq}}$.

Event 1: $R_i(\tau_k) \downarrow= \theta^a_{ii}$. In this case, $R_i(t)$ reaches the threshold θ^a_{ii} from above. It is an endogenous event and the guard condition is $g_k = R_i - \theta^a_{ii} = 0$ in (10). Therefore, the event time derivative with respect to the pq-th entry of the parameter $\boldsymbol{\Theta}$ of agent z is as follows:

$$(\tau'_k)^z_{pq} = \begin{cases} -\frac{-1+\left(R'_i(\tau_k^-)\right)^z_{pq}}{A_i - B_i N_i(\tau_k^-)} & \text{if } p = q = i, \text{ and } z = a \\ -\frac{\left(R'_i(\tau_k^-)\right)^z_{pq}}{A_i - B_i N_i(\tau_k^-)} & \text{otherwise} \end{cases} \tag{12}$$

Based on (6), this event may induce an agent departure from its current node which we denote as event DEP1. Through this event, the value of the event time derivative in (12) will be transferred to $R'_i(t)$ as shown next.

DEP1: Agent departure event 1. If Event 1 induces DEP1, then using (9) and (12), we can obtain the perturbation on node i after the agent's departure as follows:

$$\left(R'_i(\tau_k^+)\right)^z_{pq} = \left(R'_i(\tau_k^-)\right)^z_{pq} - B_i \cdot (\tau'_k)^z_{pq} \tag{13}$$

where the time perturbation $(\tau'_k)^z_{pq}$ is given by (12).

This agent departure event will induce an arrival event in the future at another target. The event time derivative in (12) will be transferred to the future arrival event, such that $\tau'_{k+l} = \tau'_k$ for some integer $l > 0$. In other words, the agent arriving at node j at τ_{k+l} carries with it the perturbation information τ'_k.

ARR1: Agent arrival event 1. This is induced by the earlier DEP1 at node i, which is again induced by the target

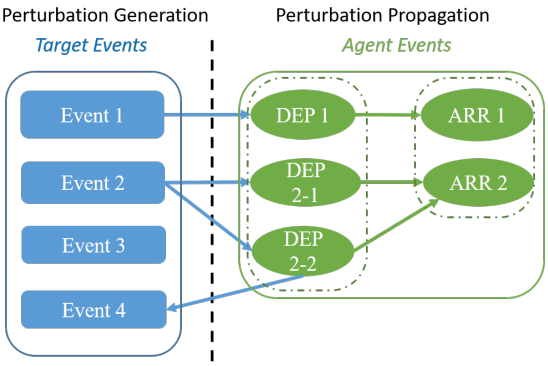


Fig. 3: Event inducing scheme and the corresponding process of perturbation generation and propagation. Blue arrows indicate instantaneous transitions if they ever occur. Green arrows indicate delayed transitions due to an agent's travel time over an edge.

event $R_i(\tau_k) \downarrow= \theta^a_{ii}$ (Event 1) and we transfer the value of the event time derivative to obtain

$$\left(\tau'_{k+l}\right)^z_{pq} = (\tau'_k)^z_{pq} \tag{14}$$

and through (9), we obtain the perturbation on node j after the agent's arrival as follows:

$$\left(R'_j(\tau_{k+l}^+)\right)^z_{pq} = \left(R'_j(\tau_{k+l}^-)\right)^z_{pq} + B_j \cdot \left(\tau'_{k+l}\right)^z_{pq} \tag{15}$$

where $(\tau'_{k+l})^z_{pq}$ is given by (14) and (12).

Event 2: $R_j(\tau_k) \uparrow= \theta^a_{ij}$. This event occurs when an agent is at node i and $R_j(t)$ at $j \neq i$ exceeds the threshold θ^a_{ij}. The event is endogenous and the guard condition in (10) is $g_k = R_j - \theta^a_{ij} = 0$. The corresponding event time derivative is obtained from (10) as follows:

$$(\tau'_k)^z_{pq} = \begin{cases} -\frac{-1+\left(R'_j(\tau_k^-)\right)^z_{pq}}{A_j - B_j N_j(\tau_k^-)} & \text{if } p = i, q = j, \text{and } z = a \\ -\frac{\left(R'_j(\tau_k^-)\right)^z_{pq}}{A_j - B_j N_j(\tau_k^-)} & \text{otherwise} \end{cases} \tag{16}$$

Looking at (6), this event can induce an agent departure event depending on whether $R_i(\tau_k) > 0$ or not: in the former case, the event is denoted by DEP2-1 and in the latter it is denoted by DEP2-2. The latter departure event DEP2-2 again has two sub-cases DEP2-2-1 an DEP2-2-2 depending on the node's dynamics after the agent's departure. The derivative in (16) will be transferred to $R'_i(t)$ through one of these agent departure events.

DEP2-1: Agent departure event 2-1. In this case, $R_i(\tau_k) > 0$. Using (9) and the event time derivative in (16), we obtain

$$\left(R'_i(\tau_k^+)\right)^z_{pq} = \left(R'_i(\tau_k^-)\right)^z_{pq} - B_i \cdot (\tau'_k)^z_{pq} \tag{17}$$

where the time perturbation $(\tau'_k)^z_{pq}$ is given in (16).

DEP2-2: Agent departure event 2-2. This event is complementary to DEP2-1 where the agent departure is induced by Event 2 but $R_i(\tau_k) = 0$. Based on (1), the target dynamics after this event either remain $\dot{R}_i(t) = 0$ or switch to $\dot{R}_i(t) = A_i - B_i N_i(\tau_k^+)$ depending on whether $A_i > B_i N_i(\tau_k^+)$ or not. Thus, there are two sub-cases to consider as follows.

DEP2-2-1: $A_i > B_i N_i(\tau_k^+)$. In this sub-case, the target dynamics switch from $\dot{R}_i(t) = 0$ for $t \in [\tau_{k-1}, \tau_k)$ to $\dot{R}_i(t) = A_i - B_i N_i(t)$ for $t \in [\tau_k, \tau_{k+1})$. We know $R_i'(\tau_k^-) = 0$ because $R_i(\tau_k) = 0$ before the agent departure and the value $R_i'(t) = 0$ holds as long as $R_i(t) = 0$. Using (9) and the event time derivative in (16), we obtain

$$\left(R_i'(\tau_k^+)\right)_{pq}^z = -(A_i - B_i N_i(\tau_k^+))\,(\tau_k')_{pq}^z \tag{18}$$

where $(\tau_k')_{pq}^z$ is again given by (16).

DEP2-2-2: $A_i \leq B_i N_i(\tau_k^+)$. In this sub-case, the target dynamics remain $\dot{R}_i(t) = 0$ before and after τ_k. Therefore, the state dynamics in (9) remain unchange and we have

$$R_i'(\tau_k^+) = 0 \quad \text{for all } p, q, z \tag{19}$$

Remark 3. Note that DEP2-2-1 induces another target event (Event 4) since $R_i(t)$ increases after the agent's departure. Moreover, both DEP2-1 and DEP2-2 will induce an agent arrival event at the next visiting target.

ARR2: Agent arrival event 2. This is induced by an earlier agent departure event at a target i which is again induced by the previous Event 2 $R_j(\tau_k) \uparrow= \theta_{ij}^a$. Similar to the derivation in **ARR1**, we transfer the prior event time derivative at τ_k to the current arrival time derivative at τ_{k+l} for some integer $l > 0$:

$$\left(\tau_{k+l}'\right)_{pq}^z = (\tau_k')_{pq}^z \tag{20}$$

Through (9) we obtain

$$\left(R_j'(\tau_{k+l}^+)\right)_{pq}^z = \left(R_j'(\tau_{k+l}^-)\right)_{pq}^z + B_j \cdot \left(\tau_{k+l}'\right)_{pq}^z \tag{21}$$

where $(\tau_{k+l}')_{pq}^z$ can be obtained by (20) and (16). Notice that the derivatives $R_j'(\tau_{k+l}^-)$ and $R_j'(\tau_k^-)$ may be different since $R_j'(t)$ may change due to arrivals or departures of other agents during $[\tau_k, \tau_{k+l})$.

Event 3: $R_i(t) \downarrow= 0$. This event corresponds to the target uncertainty state reaching zero from above, therefore from (1) the target state dynamics switch from $\dot{R}_i(t) = A_i - B_i N_i(t)$, $t \in [\tau_{k-1}, \tau_k)$ to $\dot{R}_i(t) = 0$, $t \in [\tau_k, \tau_{k+1})$. It is an endogenous event that occurs when $g_k(x, \theta) = R_i = 0$. According to (10),

$$(\tau_k')_{pq}^z = -\frac{\left(R_i'(\tau_k^-)\right)_{pq}^z}{A_i - B_i N_i(\tau_k^-)} \tag{22}$$

Replacing τ_k' in (9) with the result in (22), we have

$$\left(R_i'(\tau_k^+)\right)_{pq}^z = \left(R_i'(\tau_k^-)\right)_{pq}^z + \left(A_i - B_i N_i(\tau_k^-)\right)(\tau_k')_{pq}^z = 0 \quad \text{for all } p, q, z \tag{23}$$

This indicates that $\nabla R_i(t)$ is always reset to zero whenever the target's uncertainty state is reduced to zero. This is an uncontrollable event and does not induce any other event.

Event 4: $R_i(t) =\uparrow 0$. This event occurs when the target value leaves zero and the dynamics in (1) switch from $\dot{R}_i(t) = 0$ to $\dot{R}_i(t) = A_i - B_i N_i(t)$. It is induced by an agent departure event (DEP2-2-1) which is in turn induced by Event 2. This is an exogenous event functioning only as an indicator of $R_i(t)$ increasing from zero. Therefore, $\tau_k' = 0$ and the derivative $R_i'(t)$ will not be affected.

Remark 4. The analysis of Events 1 to 4 shows that all non-zero gradient values are caused by target events and propagated through agent departures and arrivals (see Fig.3).

IPA-based gradient descent algorithm. Using the gradient $\nabla J(\mathbf{\Theta})$ in (11), we update $\mathbf{\Theta}$ based on a standard gradient descent scheme as follows.

$$\mathbf{\Theta}^{(l+1)} = \Pi\left[\mathbf{\Theta}^{(l)} - \beta^{(l)} \nabla J(\mathbf{\Theta}^{(l)})\right] \tag{24}$$

where the operator $\Pi \equiv \max\{\cdot, \mathbf{0}\}$, (l) indexes the number of iterations, and $\beta^{(l)}$ is a diminishing step-size sequence.

IV. One-agent case analysis

Recalling our control policy in (6), the diagonal entries of $\mathbf{\Theta}$ control the dwell times at nodes, whereas the off-diagonal entries control the feasible node visiting sequence. We ignore the superscript agent index in single-agent cases and show that the optimal values of diagonal entries are always zero. This structural property indicates that the agent visiting a node should always reduce the uncertainty state to zero before moving to the next node.

Assumption 1. For any $\epsilon > 0$, there exists a finite time horizon $T > t_K - \frac{c}{1-\epsilon}$ where t_K is an instant such that $\|\nabla R_i(t_1) - \nabla R_i(t_2)\| \leq \epsilon/M$, $i = 1, \ldots, M$ for all $t_1, t_2 > t_K$ and c is a finite constant.

Assumption 2. The node visiting sequence is fixed.

The first assumption is a technical one which ensures that the optimization problem is defined over a sufficiently long time horizon to allow the gradient to converge so as to obtain Θ_d^* in Theorem 1 below. The second assumption allows us to reduce the parameter matrix Θ to a vector of its diagonal elements only $\Theta_d = [\theta_1, \theta_2, \ldots, \theta_M]^\top \geq \mathbf{0}_{M\times 1}$.

Theorem 1. *Consider M targets and a single agent under the parametric control Θ_d. Under Assumptions 1 and 2, the optimal thresholds satisfy $\Theta_d^\star = \mathbf{0}_{M\times 1}$.*

Remark 5. The result of Theorem 1 is consistent with, but more general than, a similar result in [12] where homogeneous targets are assumed. Moreover, our proof in [13] shows that if T is sufficiently large and ϵ is arbitrarily small, $\nabla R_i(t)$ will converge to: $\partial R_i/\partial\theta_i = 1$ and $\partial R_i/\partial\theta_j = 0$ for $j \neq i$ for every node $i = 1, \ldots, M$ and $\lim_{T\to\infty} \partial J(\Theta_d)/\partial\theta_i = 1$.

V. Simulation Examples

One agent, two targets. We provide this simple example to illustrate Theorem 1. Consider a controller with parameter vector $\Theta_d = [\theta_1, \theta_2]^\top$. We track the evolution of $\nabla R(t) = \left[\frac{\partial R_1}{\partial\theta_1}, \frac{\partial R_1}{\partial\theta_2}, \frac{\partial R_2}{\partial\theta_1}, \frac{\partial R_2}{\partial\theta_2}\right]^\top$ by events. The agent is initialized at node 1. Its trajectory is cycles with each cycle consisting of four stages: i) the agent's departure from node 1, ii) arrival at node 2, iii) departure from node 2, and iv) return to node 1. Using the IPA results in Sec. III, we combine the evolution of $\nabla R(t)$ from stage i) to iv) and use t_k to denote the beginning of the k-th cycle, and obtain

$$\nabla R(t_{k+1}) = \Lambda \nabla R(t_k) + U \tag{25}$$

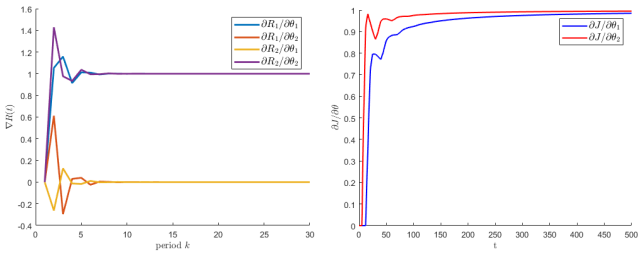


Fig. 4: Left: convergence of $\nabla R(t)$ to the equilibrium $[1,0,0,1]^\top$. Right: convergence of $\partial J/\partial\Theta_d$ to $[1,1]^\top$.

where Λ and U are the combined update matrix and vector (see details in [13]). The only equilibrium of the system defined by (25)is $\nabla R_e = (I-\Lambda)^{-1}U = [1,0,0,1]^\top$. $\nabla R(t)$ converges to that equilibrium asymptotically. The results (Fig.4) match our analysis in Remark 5. Through (24), Θ_d is eventually reduced to $[0,0]^\top$.

Multi-agent cases: a counterexample to Theorem 1. The nice property in Theorem 1 does not generally apply to multi-agent cases. This is because when agents are visiting a node, the allocation of agents to nodes may be improved if one agent leaves the node before reducing its uncertainty to zero and allow other agents to complete the task. Due to space limitations, settings of this counterexample including the initial and final thresholds are listed in [13]. The diagonal entries of the final parameter matrices for both agents are: $\Theta_d^{1*} = [0,0,7.25,8.06,0.23]^\top$ and $\Theta_d^{2*} = [0.88,19.27,0.02,0.01,0]^\top$ which do not satisfy the structure given in Theorem 1. The node visiting sequences and dwell times are optimized on-line. However, since agents may adjust their visiting sequences asynchronously, the cost in multi-agent cases fluctuates during the optimization process.

Threshold-based policy versus dynamic programming. We present an example consisting of 1 agent and 4 nodes to compare the performance of the threshold-based policy with a dynamic programming solution using value iteration. Details can be found in [13]. Using dynamic programming, the value function converges after 15 iterations and the final cost $J^\star_{\text{DP}} = 31.15$. However, the running time is about 16 minutes per value iteration using a computer with Intel(R) Core(TM) i7-7700 CPU @3.60GHZ processor. On the other hand, the solution obtained by the threshold-based IPA approach results in a slightly higher cost, but the computational complexity is reduced by several orders of magnitude as shown in Fig. 5. After about 30 seconds in total running time on the same computer, the cost is reduced to $J^\star_{\text{IPA}} = 36.20$.

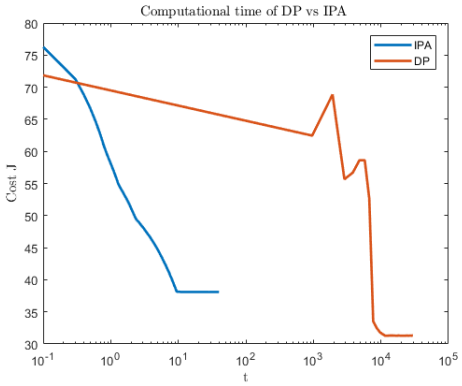


Fig. 5: Cost versus computational time (in log scale). The blue line shows the result of IPA with the final cost $J^\star_{\text{IPA}} = 36.20$ and the orange line shows the result of dynamic programming with the final cost $J^\star_{\text{DP}} = 31.15$.

VI. Conclusions

The multi-agent persistent monitoring problem involves the planning of agent trajectories defined both by the sequence of nodes (targets) to be visited and the amount of time spent at each node. We have considered a class of distributed parametric controllers through which agents control their visit sequence and dwell times at nodes using threshold parameters associated with the node uncertainty. We have used Infinitesimal Perturbation Analysis (IPA) to determine online optimal threshold parameters through gradient descent and thus obtain optimal controllers within this family of threshold-based policies. Compared with dynamic programming solutions, our threshold-based parametric controller reduces the computational time substantially by orders of magnitude. In future work, richer families of threshold-based controllers will be developed by considering multi-step-look-ahead policies and by identifying structural properties therein which show the trade-off between exploitation and exploration in persistent monitoring tasks.

References

[1] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proc. American Control Conference*, 2005, pp. 1859–1864.

[2] M. Zhong and C. G. Cassandras, "Distributed coverage control and data collection with mobile sensor networks," *IEEE Trans. on Automatic Control*, vol. 56, no. 10, pp. 2445–2455, 2011.

[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] 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.

[5] T. T. Ashley, E. L. Gan, J. Pan, and S. B. Andersson, "Tracking single fluorescent particles in three dimensions via extremum seeking," *Biomedical optics express*, vol. 7, no. 9, pp. 3355–3376, 2016.

[6] S. M. Cromer Berman, P. Walczak, and J. W. Bulte, "Tracking stem cells using magnetic nanoparticles," *Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotech.*, vol. 3, no. 4, pp. 343–355, 2011.

[7] 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, Apr. 2012.

[8] N. Zhou, X. Yu, S. B. Andersson, and C. G. Cassandras, "Optimal event-driven multi-agent persistent monitoring of a finite set of targets," in *Proc. IEEE Conf. on Decision and Contr.*, 2016, pp. 1814–1819.

[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 Automatic Contr.*, vol. 60, no. 6, pp. 1659–1664, 2015.

[10] N. Zhou, C. G. Cassandras, X. Yu, and S. B. Andersson, "Optimal event-driven multi-agent persistent monitoring with graph-limited mobility," in *Proc. IFAC World Congress*, 2017, pp. 2181–2186.

[11] 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.

[12] X. Yu, S. B. Andersson, N. Zhou, and C. G. Cassandras, "Optimal dwell times for persistent monitoring of a finite set of targets," in *Proc. American Control Conference*, 2017, pp. 5544–5549.

[13] N. Zhou, C. G. Cassandras, X. Yu, and S. B. Andersson, "Optimal threshold-based control policies for persistent monitoring on graphs," *preprint arXiv:1803.02798*, 2018.