Available online at www.sciencedirect.com
ScienceDirect
Digital Signal Processing 16 (2006) 546–567
Digital Signal Processing
www.elsevier.com/locate/dsp

Adaptive multi-modality sensor scheduling for detection and tracking of smart targets

Chris Kreucher [a], Doron Blatt [a], Alfred Hero [a,*], Keith Kastella [b]

[a] *Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122, USA*

[b] *General Dynamics Advanced Information Systems, Ann Arbor, MI 48113-4008, USA*

Available online 18 January 2005

Abstract

This paper considers the problem of sensor scheduling for the purposes of detection and tracking of "smart" targets. Smart targets are targets that can detect when they are under surveillance and react in a manner that makes future surveillance more difficult. We take a reinforcement learning approach to adaptively schedule a multi-modality sensor so as to most quickly and effectively detect the presence of smart targets and track them as they travel through a surveillance region. An optimal scheduling strategy, which would simultaneously address the issue of target detection and tracking, is very challenging computationally. To avoid this difficulty, we use a two stage approach where targets are first detected and then handed off to a tracking algorithm. We investigate algorithms capable of choosing whether to use the active or passive mode of an agile sensor. The active mode is easily detected by the target, which makes the target prefer to move into hide mode. The passive mode is nearly undetectable to the target. However, the active mode has substantially better detection and tracking capabilities then the passive mode. Using this setup, we characterize the advantage of a non-myopic policy with respect to myopic and random polices for multitarget detection and tracking.

Keywords: Multistage sensor management; Non-myopic scheduling; Reinforcement learning; Target detection; Target tracking

* Corresponding author.
E-mail addresses: ckreuche@umich.edu (C. Kreucher), dblatt@umich.edu (D. Blatt), hero@umich.edu (A. Hero), keith.kastella@gd-ais.com (K. Kastella).

1051-2004/$ – see front matter
doi:10.1016/j.dsp.2004.12.008

1. Introduction

The problem of sensor scheduling is to determine the best way to task a sensor or group of sensors when each sensor may have many modes and search patterns. Tasking a sensor may include such choices as where to point, what mode to use, and what signal to transmit. In general, sensors must balance complex tradeoffs between competing mission goals, e.g., detection of new targets, tracking of existing targets, and identification of existing targets.

An optimal sensor scheduling algorithm will depend on the posterior distribution of the system state conditioned on sensor measurements. In our application, the system state describes probabilistically both the uncertainty in number of targets and locations of the individual targets. In principle, one could derive an optimal scheduling algorithm that simultaneously treats detection of new targets and tracking of existing targets by defining an appropriate global reward. However, in practice, this is very difficult due to computational considerations. To combat this challenge, in this paper we take a modular approach and treat the problem in two stages—target detection followed by target tracking. This suboptimal algorithm can be viewed as an approximation to an optimal algorithm which simultaneously considers detection and tracking.

Sensor scheduling is complicated substantially when targets under surveillance are able to detect and respond to sensing activities (so called "smart" targets). In this paper, we consider such a scenario. Specifically, we investigate the situation where a sensor is charged with detecting and tracking a group of moving ground targets and the targets have the ability to detect some of the surveillance actions and respond by concealing their whereabouts.

Operationally, we envision an adversarial target proceeding along some terrain amenable to traveling. Upon detecting surveillance activity, the target will tend to move off the good terrain to less hospitable areas (e.g., among the trees so as to be under foliage). This area is less desirable to the target than the good terrain as it may be more difficult to traverse or be more dangerous (e.g., due to the fact that the area is unsurveyed, it may contain mud, ditches or other obstacles that immobilize the target). However, this less hospitable area is beneficial as it obscures the target, combating future surveillance attempts. The target tends to stay in the less hospitable area until it has high confidence that the region is no longer under surveillance and then moves back to the hospitable area to continue its journey.

The sensor must trade among several modalities to most quickly and effectively detect and track the targets. We consider the case where the sensor has an active mode and a passive mode. The active mode has good performance characteristics in terms of detection rate and false alarm probability, while the passive mode has reduced performance characteristics. The active mode, however, suffers from the fact that it is easily detectable by the target (causing the target to go into hide mode) whereas the passive mode is nearly undetectable. Therefore, the sensor scheduling algorithm is faced with the tradeoff between using a high quality sensor, which may damage future sensing ability, versus using a poorer quality sensor which leaves future sensing ability intact.

Sensor scheduling strategies may be myopic or non-myopic. In the myopic case, sensing actions are taken so as to maximize the immediate reward. Myopic methods have

the advantage that they are more computationally tractable than non-myopic methods. Many researchers have investigated myopic methods of sensor management, including Refs. [8,13,17,21].

On the other hand, a full non-myopic solution takes into account the future benefit (or cost) of current actions to maximize long term payoff. Non-myopic methods are often formulated with a Markov decision process (MDP) strategy. However, the long-term (non-myopic) scheduling solution suffers from combinatorial explosion when solving practical problems of even moderate size. Researchers have worked at approximate solution techniques. For example, Krishnamurthy [11,12] uses a multi-arm bandit formulation involving hidden Markov models. In Ref. [12], an optimal algorithm is formulated to track multiple targets with an electronically scanned array that has a single steerable beam. Since the optimal approach has prohibitive computational complexity, several suboptimal approximate methods are given. Bertsekas and Castanon [5] formulate heuristics for the solution of a stochastic scheduling problem corresponding to sensor scheduling. They implement a rollout algorithm based on their heuristics to approximate the stochastic dynamic programming algorithm.

Another approach to long term decision making is reinforcement learning (RL) [18]. In this approach, training examples of sensing actions, responses, and the observed system states are used to learn an optimal sensor scheduling policy.

Sensor scheduling to detect and track smart targets is an application that strongly benefits from non-myopic decision making. In the target tracking setting, a myopic strategy would choose to always use the active mode. A non-myopic strategy might choose to use the passive mode for some time and then switch to the active mode after establishing a certain level of confidence about the target. In this paper, we investigate a RL approach to the sensor scheduling problem. Although at first it may seem that a RL approach will be difficult to implement in practice, as training examples are not usually available, there are good reasons for its investigation. First, if models and simulations of the battlefield environment exist, the RL algorithm can be trained to provide a good policy in the laboratory. When deployed it may be possible to continually update policies based on real data, thereby continually improving on the policy. Also, the RL strategy is a useful way to establish a bound on the best possible performance in complicated situations such as the one we investigate here. This bound can be used to judge the quality of approximate strategies in terms of their closeness to optimal.

We use RL to detect and track targets in a two step approach. First, we use a detection algorithm trained to most quickly decide on the presence or absence of smart targets in a portion of the surveillance region. During training, the correct-decision reward is decreased over time to encourage quickest detection. Once a target is detected, the tracking algorithm is initiated and tipped off to the presence of targets. The tracking algorithm has the responsibility to finely geolocate and track the targets as they move through the region. The tracking algorithm is based on a combination of RL techniques with an information theoretic reward function. We show in the smart target problem this two stage approach provides an effective method of deciding what sequence of sensor modes to deploy.

The paper proceeds as follows. Section 2 outlines our two stage detection and tracking algorithm. Second, Section 3 gives an overview of RL methods and specifically Q-learning. Third, in Section 4, we describe the application of RL to the two stage de-

tection and tracking algorithm. Fourth, in Section 5, we provide simulation results of the algorithm for two smart targets. The method is compared to random and myopic strategies and shown to provide good performance. Finally, in Section 6 we conclude with some summarizing remarks.

2. Overview of the approach

We decouple the scheduling problem into two disjoint optimization problems, as illustrated by Fig. 1. This factorization approach is suboptimal, but allows for development of a more computationally tractable algorithm than solving the detection and tracking problem jointly.

The first stage of the algorithm is a quickest detection problem, which attempts to determine the presence or absence of targets in the surveillance region. The detection stage proceeds by dividing the surveillance region into a set of coarse detection regions. For each

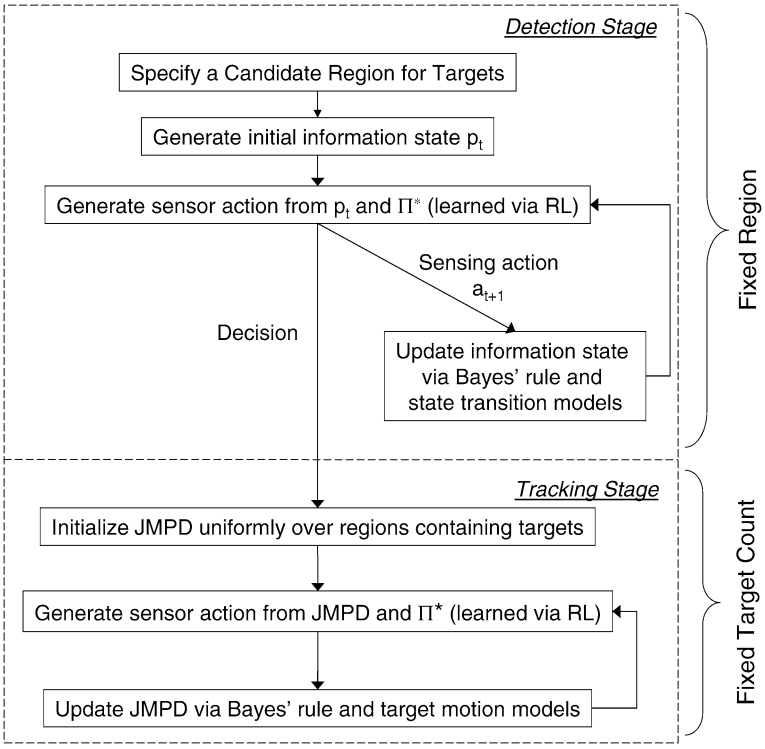

Fig. 1. An illustration of the two stage approach to smart target detection and tracking. The detection stage performs sensor scheduling to most quickly decide upon the presence or absence of targets in a detection region. Upon determining that a target is present, the tracking algorithm is responsible for scheduling sensors to finely locate and track the moving targets. Both stages of the algorithm rely on reinforcement learning (RL) where the best sensor scheduling strategy is learned. Here the information space $\mathbf{p}_t$ is the posterior distribution of target states at time t and Π^* denotes the optimal policy (mapping from $\mathbf{p}_t$ to sensing actions a_{t+1}).

of these detection regions, sensing actions are performed so as to most quickly decide on the state of the region.

When a target (or targets) are detected, the tracking algorithm is initiated. The tracking algorithm uses the coarse prior information provided by the detection stage to initialize the tracker. Sensing actions are then performed to finely geolocate and track the targets.

3. Reinforcement learning for optimal solution of a MDP

In this section, we give an overview of the mathematics used to address the problem of smart target detection and tracking. We use an infinite horizon Markov decision process (MDP) [15] to mathematically characterize the problem. The main challenge one faces in finding MDP solutions is that the complexity of finding optimal policies grows exponentially with the state and action spaces [4]. Since the sensor scheduling problem is characterized by extremely large state and action spaces, it is necessary to develop approximate solutions using dimension reduction. We advocate methods from reinforcement learning (RL) coupled with function approximation to find approximately optimal policies for the two stages.

3.1. Infinite-horizon MDP

A discounted-reward infinite-horizon MDP is defined by a sequence of states $\{S_t\}_{t\geqslant 0}$ taking values in a state space $\mathcal{S}$, a sequence of actions $\{A_t\}_{t\geqslant 0}$ taking values in an action space $\mathcal{A}$, and a (possibly random) reward function $r(S_t, A_t)$ that assigns the cost incurred (when negative) or the reward gained (when positive) to the event of being at state S_t and taking action A_t. In our context, the state space characterizes the battlefield. It contains rich information such as the number of targets present, their location, their type, and whether they are stationary or moving. The action space contains all the possible actions. Each action specifies which sensor to use, the mode of operation, and where to point the sensor. The reward system reflects the tradeoffs between costs of deploying a certain sensor mode and the gain earned from the measurement it collects.

The MDP is initiated with state S_0 followed by action A_0 chosen by the controller and continues with the state-action sequence $S_1, A_1, S_2, A_2, \ldots$. Under the Markovian model, given S_t and A_t, S_{t+1} is independent of all past states and actions. The state transitions are governed by a stationary probabilistic law, denoted by $p(S_{t+1}|S_t, A_t)$, that specifies the distribution of S_{t+1} over $\mathcal{S}$, given S_t and A_t. $p(S_{t+1}|S_t, A_t)$ is either a probability density function when the state space is continuous or a probability mass function when it is discrete.

A stationary policy Π is a map from $\mathcal{S}$ to $\mathcal{A}$ that specifies the action taken at each state. Denote the class of all policies by $\mathcal{P}$. The value function associated with policy Π, denoted by $V^{\Pi}(s)$ is the expected total discounted reward when in state $S_t = s$ and following policy Π, that is

$$V^{\Pi}(s) = E\left\{\sum_{\tau=t}^{\infty} \beta^{\tau-t} r\big(S_\tau, \Pi(S_\tau)\big)\big|S_t = s\right\}, \quad \forall s \in \mathcal{S}, \tag{1}$$

where $\beta \in (0, 1)$ is a discount factor, which is included to reduce the value of future rewards as compared with immediate rewards. The conditional expectation is taken with respect to the joint distribution of all the targets, which, in the context of smart targets, is highly dependent on the action sequence. Therefore, a direct calculation of this expression is computationally intractable. An optimal policy is a policy that satisfies

$$\Pi^*(s) = \arg\max_{\Pi \in \mathcal{P}} V^{\Pi}(S), \quad \forall s \in \mathcal{S}. \tag{2}$$

It is well known that the optimal policy is the unique solution to Bellman's equation

$$V(s) = \max_a E\big\{r(S_t, a) + \beta V(S_{t+1}) | S_t = s,\ A_t = a\big\}, \tag{3}$$

and can be found using Bellman's iterations [15]: given an arbitrary value function $V_1(s)$, the sequence generated by

$$V_{k+1}(s) = \max_a E\big\{r(S_t, a) + \beta V_k(S_{t+1}) | S_t = s,\ A_t = a\big\} \tag{4}$$

converges to $V^*(s)$, that is, $\lim_{k\to\infty} V_k(s) = V^*(s)$. Given $V^*(s)$, the optimal actions in Π^* are computed by

$$\arg\max_a E\big\{r(S_t, a) + \beta V^*(S_{t+1}) | S_t = s,\ A_t = a\big\}. \tag{5}$$

Unfortunately, when the state and action spaces are large and the state transition density is either computationally complicated or not explicitly available, this method is intractable and one must use approximate methods such as Q-learning [4].

3.2. *Q-Learning*

The optimal scheduling policy for the two stages is found using Q-learning coupled with function approximation [18–20]. The learning part relaxes the requirement for explicit knowledge of the transition density, and function approximation is used to further reduce the dimensionality of the state and action spaces.

Given the value function V^*, the Q-function is defined by

$$Q(s, a) = E\big\{r(s, a) + \beta V^*(S_{t+1}) | S_t = s,\ A_t = a\big\}, \tag{6}$$

i.e., the expected reward when taking action a at state s and then acting optimally for all future actions. The Q-function satisfies the equation

$$Q(s, a) = E\big\{r(s, a) + \beta \max_{\alpha \in \mathcal{A}} Q(S_{t+1}, \alpha) | S_t = s,\ A_t = a\big\}. \tag{7}$$

Given the Q-function, actions are computed as

$$\arg\max_{a \in \mathcal{A}} Q(S_t, a). \tag{8}$$

In Q-learning, the Q-function is estimated from multiple realizations of the state-action sequence. Specifically, the training process involves generation of {*state, action, next state, immediate reward*} 4-tuples over a large number of training episodes. In our approach, this set of training episodes is used in batch to determine the Q-function for a particular state-action pair. Specifically, assume that both $\mathcal{S}$ and $\mathcal{A}$ are finite. Then, there exists a lookup

table representation of $Q(s,a)$. In this case, given an arbitrary initial value of $Q(s,a)$, the one-step Q-learning algorithm [18] is given by the repeated application of the update equation

$$Q_k(s,a) = (1-\gamma)Q_{k-1}(s,a) + \gamma\big(r + \beta \max_{\alpha\in\mathcal{A}} Q_{k-1}(s',\alpha)\big), \tag{9}$$

where each of the 4-tuples $\{S_t = s, A_t = a, S_{t+1} = s', R_t = r\}$ are incurred during the progress of the MDP, and $\gamma \in (0,1)$ decreases with t. This algorithm can be seen as the Robbins–Monro stochastic approximation method for solving (Eq. (7)). Therefore, when γ decreases to zero as $a/(b+t)$ for some positive constants a and b, this algorithm converges to the true Q-function with probability 1 regardless of the actual policy used in generating the trajectories as long as the state action pairs are visited infinitely often [4,18].

Unfortunately, in most realistic problems (the problems discussed herein included) it is infeasible to represent the Q-function in a lookup table, either because the number of states is too large or simply because the state space is continuous. Therefore, we require a function approximation technique to represent the Q-function. Less is known about the convergence properties of Q-learning with function approximation, and in practice its properties depend on the policy used to generate the trajectories and the function approximation class (see the discussion in Ref. [4]). The standard and simplest class of Q-function approximators are linear combinations of basis functions (also called features), i.e.,

$$Q(s,a) = \theta^T \phi(s,a), \tag{10}$$

where $\phi(s,a)$: $\mathcal{S}\times\mathcal{A}\to\mathbb{R}^L$ is a feature vector associated with state s and action a and the coefficients of $\theta\in\mathbb{R}^L$ are to be estimated by $\hat{\theta}$, i.e., the training data is used to learn the best approximation to $Q(s,a)$ among all linear combinations of the features. Choosing a feature vector $\phi(s,a)$ to represent the state is a challenging problem that will be addressed separately for each of the two stages of the algorithm.

We use a gradient descent method [18] for updating the Q-function with new data. Under the function approximation (Eq. (10)) for Q, this amounts to estimating the parameter vector θ using the received training data which consists of an observed state, a chosen action, an observed reward and an observed next state, $\{s,a,r,s'\}$:

$$\begin{aligned}\hat{\theta} &\leftarrow \hat{\theta} + \gamma\big(r + \beta\max_{a'} Q(s',a') - Q(s,a)\big)\nabla_\theta Q(s,a)\\ &= \hat{\theta} + \gamma\big(r + \beta\max_{a'} \hat{\theta}^T\phi(s',a') - \hat{\theta}^T\phi(s,a)\big)\phi(s,a),\end{aligned}$$

where $\gamma\in(0,1)$ decreases with t. Hence, at every iteration, $\hat{\theta}$ is updated in the direction that minimizes the empirical error in Eq. (7). When a lookup table is used in Eq. (10), this algorithm reduces to Eq. (9). Once the learning of the vector θ is completed, actions are computed according to

$$\arg\max_{a\in\mathcal{A}} \hat{\theta}^T\phi(S_t,a). \tag{11}$$

3.3. Partially observable Markov decision processes

In some situations the true state of the system is unknown to the controller. Instead, only noisy measurements of the system's state are available. In this case, even if the state transition probability law is Markovian, the noisy measurements are no longer Markovian. To overcome is difficulty, the process is reformulated in terms of the information state [3]. This formalization leads to a partially observable Markov decision process (POMDP) that can be handled using the framework described above, because with the concept of an information state all POMDP's can be recast into the MDP framework.

The information state is defined as the posterior distribution of states given all past measurements, $p(s|\mathbf{Z}_t)$, where $\mathbf{Z}_t$ denotes all past measurements up to and including time t, i.e., $\mathbf{Z}_t = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t\}$, where $\mathbf{z}_t$ is the measurement collected at time t. Note that the term information state is unrelated to the information theoretic measures used in the reward described later. The information state is a sufficient statistic for the problem in the sense that the expected reward depends on the data only through the information state. Denote by $p_0(s)$ the prior distribution of the states which constitutes the initial information state. In our setting, this information corresponds to prior intelligence on the surveillance region. In the absence of such information, a uniform distribution or other non-informative distribution over the state space can be used. At every stage of the process, given the current information state $p_t(s)$ and a new measurement $\mathbf{z}_{t+1}$ the next information state $p_{t+1}(s)$ is computed by Bayes rule

$$p_{t+1}(s) \stackrel{\triangle}{=} p(s|\mathbf{Z}_{t+1}) = p(s|\mathbf{Z}_t, \mathbf{z}_{t+1}) = \frac{p_t(s) f(\mathbf{z}_{t+1}|s)}{\sum_{\mathcal{S}} p_t(s) f(\mathbf{z}_{t+1}|s)}, \quad \forall s \in \mathcal{S}, \tag{12}$$

where $f(\mathbf{z}_{t+1}|s)$ denotes the conditional density of the measurement $\mathbf{z}_t$ given the true state of the system is s, and we assume that given the true system state the measurements are independent. Hence, a new information state depends on the past measurements only though the previous information state. This formalization leads to a POMDP with a continuous state space, which is the space of all probability vectors over the unknown system states.

The methods available in the literature for finding optimal policies in the POMDP setting focus on the case of finite observation and action spaces and finite horizon problems [1,14]. The quickest detection problem with continuous observation space discussed below does not fall into this class of problems. Therefore, Q-learning coupled with function approximation can be used to approximate the optimal policy.

4. Application of RL to detection and tracking of smart targets

In this section we present the details of the application of Q-learning to the two stages of the multitarget detection and tracking algorithm.

4.1. Detecting smart targets

The target detection stage is formulated as a Bayesian hypothesis testing problem in which one is trying to decide between $M \geqslant 2$ hypotheses: $\mathrm{H}_1, \ldots, \mathrm{H}_M$. The observed system is modelled as a MDP with a finite state space $\mathcal{S}$ with cardinality N. Each hypothesis

corresponds to a different subset of the states and it is assumed there are no transitions between states that are associated with different hypotheses. For example, H_1 can correspond to the hypothesis that a target is not present and H_2 to the hypothesis that a target is present. Under H_1 the system has only one state, and under H_2 the target can be at one of several states that determine if the target is hidden or exposed. The target can have state transitions under H_2 but cannot switch between H_1 and H_2.

At each time instant t, one of K sensor modes denoted by $\Sigma_1, \ldots, \Sigma_K$ is used to collect a measurement $\mathbf{z}_t$, or alternatively a final decision is made. Therefore, the possible actions available at each time epoch are $\mathcal{A} = \{\Sigma_1, \ldots, \Sigma_K, D\}$, where D stands for the action of making the final decision. After action D the detection process ends and a reward is granted for a correct decision.

Denote by $f_k(\mathbf{z}|s)$ the conditional density of a measurement collected by sensor mode k given the system is at state s. The state transition probabilities of the Markov process $p(S_{t+1}|S_t, A_t)$ depend on the deployed sensor mode. The possible states in $\mathcal{S}$ are enumerated from 1 to N and the transition probabilities are summarized in the matrices $\mathbf{A}_k$, $k = 1, \ldots, K$, where

$$[\mathbf{A}_k]_{nl} = p(S_{t+1} = n | S_t = l, \Sigma_k), \quad n, l = 1, \ldots, N \tag{13}$$

is the probability that the system moves from state l to state n when sensor mode k is used.

The dependency on the deployed sensor mode is applicable when a target can sense it is being observed and may react accordingly, e.g., hide. If mode k is deployed, cost c_k is incurred, i.e., $r(s, \Sigma_k) = -c_k$ for all s. If a correct decision is made, reward R is received. If an erroneous final decision is made, no reward is received, i.e., for all s, $r(s, D) = R$ when the decision is correct and $r(s, D) = 0$ otherwise. As described in Section 3.3, $\mathbf{Z}_t$ denotes the information available to the system at time t, which includes measurements collected up to time t. Since the number of states is finite and known, we use the vector notation $\mathbf{p}_t$ to denote the posterior probability vector of target states given $\mathbf{Z}_t$. Using this notation, if sensor k is deployed and $\mathbf{z}_{t+1}$ is collected, Eq. (12) takes the form

$$\mathbf{p}_{t+1} = \frac{\mathbf{A}_k \operatorname{diag}([f_k(\mathbf{z}_{t+1}|1), \ldots, f_k(\mathbf{z}_{t+1}|N)])\mathbf{p}_t}{\operatorname{sum}(\mathbf{A}_k \operatorname{diag}([f_k(\mathbf{z}_{t+1}|1), \ldots, f_k(\mathbf{z}_{t+1}|N)])\mathbf{p}_t)}, \tag{14}$$

where $f_k(\mathbf{z}_{t+1}|n)$ denotes the conditional density of a measurement that was collected by sensor k given that the system is in state n, and for any vector $\mathbf{v}$, $\operatorname{diag}(\mathbf{v})$ is a diagonal matrix with the elements of $\mathbf{v}$ on its diagonal, and $\operatorname{sum}(\mathbf{v})$ is the sum of its elements. Therefore, a policy $\Pi \in \mathcal{P}$ can be defined as a map from $\mathcal{S}^N$, the simplex of N-dimensional probability vectors, to $\mathcal{A}$. The expected total reward at information state $\mathbf{p}_t$ associated with a policy becomes

$$V^{\Pi}(\mathbf{p}_t) = E\left\{\sum_{\tau=t}^{\infty} \beta^{\tau-t} r\big(\mathbf{p}_\tau, \Pi(\mathbf{p}_\tau)\big)\right\}, \tag{15}$$

and the optimal policy is

$$\Pi^* = \arg\max_{\Pi \in \mathcal{P}} V^{\Pi}(\mathbf{p}), \quad \forall \mathbf{p} \in \mathcal{S}^N. \tag{16}$$

The Q-function is defined over the N-dimensional simplex $\mathcal{S}^N$ and for any action $a \in \mathcal{A}$ by

$$Q(\mathbf{p}_t, a) = E\big\{r(\mathbf{p}_t, a) + \beta V^*(\mathbf{p}_{t+1})\big\}, \tag{17}$$

which is the expected reward when taking action a at information state $\mathbf{p}_t$ and then acting optimally thereafter. The dimensionality of the information state space is reduced by a linear parametrization (Eq. (10)), and Q-learning is used to approximate the Q-function. Given Q, one finds the optimal policy by taking the action that maximizes it at any given information state.

4.2. Tracking smart targets

Tip-offs from the detection algorithm are used to initialize a tracking algorithm which finely geolocates and tracks moving targets. Targets are tracked by recursively estimating a conditional probability density known as the joint multitarget probability density (JMPD) [9,10].

4.2.1. The JMPD and particle filter approximation

In the tracking stage, the state s of the system (see Section 4) is given by the joint multitarget probability density. In this subsection, we show how the state is derived and how states are combined with measurements to determine the next state.

We define the joint multitarget conditional probability density $p(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^{T-1}, \mathbf{x}_t^T | \mathbf{Z}_t, T_t)$ as the probability for T targets with states $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{T-1}, \mathbf{x}^T$ at time t based on observations $\mathbf{Z}_t$. Each of the state vectors $\mathbf{x}^i$ in the JMPD is a vector quantity and may (for example) be of the form $[x, \dot{x}, y, \dot{y}]'$. For convenience, the density will be written compactly as

$$p(\mathbf{X}_t, T_t | \mathbf{Z}_t) = p\big(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^{T-1}, \mathbf{x}_t^T | \mathbf{Z}_t\big), \tag{18}$$

where $\mathbf{X}_t = [\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^{T-1}, \mathbf{x}_t^T]$.

The temporal update of the posterior likelihood on this density proceeds according to the usual rules of Bayesian filtering. Given a kinematic model of how the JMPD evolves over time, $p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{X}_t, T_t)$, we compute the time-updated prediction density via marginalization of a conditional density:

$$p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{Z}_t) = \sum_{T_t=0}^{\infty} \int d\mathbf{X}_t \, p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{X}_t, T_t) \, p(\mathbf{X}_t, T_t | \mathbf{Z}_t). \tag{19}$$

$p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{Z}_t)$ is referred to as the prior or prediction density at time $t+1$, as it is the density at time $t+1$ conditioned on measurements up to and including time t.

Given a sensor model, $p(\mathbf{z} | \mathbf{X}_t)$, Bayes' rule is used to update the posterior density as a new measurement vector $\mathbf{z}$ arrives at time $t+1$ via

$$p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{Z}_{t+1}) = \frac{p(\mathbf{z} | \mathbf{X}_{t+1}) \, p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{Z}_t)}{p(\mathbf{z} | \mathbf{Z}_t)}. \tag{20}$$

$p(\mathbf{X}_{t+1}, T_{t+1} | \mathbf{Z}_{t+1})$ is referred to as the posterior or the updated density at time $t+1$ as it is the density at time $t+1$ conditioned on all measurements up to and including

time $t+1$. Note that, in contrast to the detection framework, here we must estimate a high-dimensional joint probability density. This density allows us to describe the uncertainty about the precise location of the target (rather than simply a region) and to represent the correlations that occur due to measurement ambiguity for multiple targets.

The sample space of $\mathbf{X}$ is very large. It contains all possible configurations of state vectors $\mathbf{x}^i$. We find that a particle filter based representation of the JMPD allows tractable implementation [9]. The particle filter approximation represents the JMPD by a collection of weighted Dirac samples, i.e.,

$$p(\mathbf{X}, T|\mathbf{Z}) \approx \sum_{p=1}^{N_{\text{part}}} w_p \delta(\mathbf{X} - \mathbf{X}_p). \tag{21}$$

Particle filtering is a method of approximately solving the prediction and update equations (19) and (20) by simulation [6]. Samples are used to represent the density and to propagate it through time. The prediction equation (19) is implemented by proposing new particles from the existing particles using a model of state dynamics and the measurements. The update equation (20) is implemented by assigning a weight to each of the particles that have been proposed using the measurements and the model of state dynamics.

We use an adaptive method of particle proposal [9] that automatically factorizes the JMPD when permissible. This adaptive sampling method automatically determines the most efficient particle proposal method allowing tractable implementation for tens of targets.

4.2.2. Information based myopic sensor management

We use the JMPD to make sensor tasking decisions. As others have realized [8,13,21], a good measure of the quality of a sensing action is the reduction in entropy of the posterior distribution induced by the measurement. Therefore, the reward of an action (Section 4) will be given by the information gained by taking that action. To schedule a sensor, we enumerate all possible sensing actions (e.g., sensor modes and sensor pointing directions) and calculate the *expected* gain in information associated with each possible action.

The calculation of information gain between two densities f_1 and f_0 is done using the Rényi information divergence [7,16], also known as the α-divergence:

$$D_\alpha(f_1||f_0) = \frac{1}{\alpha - 1} \ln \int f_1^\alpha(x) f_0^{1-\alpha}(x)\, dx. \tag{22}$$

The α-divergence includes the Kullback–Leibler divergence (as $\alpha \to 1$) and is related to the Hellinger distance at $\alpha = 0.5$ [7]. There is both theoretical and empirical evidence suggesting that $\alpha = 0.5$ is appropriate for the tracking problem [7,10], and it is used in all simulations reported in this paper.

In our application, we are interested in computing the divergence between the predicted density $p(\mathbf{X}_{t+1}|\mathbf{Z}_t)$ and the updated density, $p(\mathbf{X}_{t+1}|\mathbf{Z}_{t+1})$. Particle filter approximation of the density simplifies Eq. (22) to

$$D_\alpha\big(p(\cdot|\mathbf{Z}_{t+1})||p(\cdot|\mathbf{Z}_t)\big) = \frac{1}{\alpha - 1} \ln \frac{1}{p(\mathbf{z})^\alpha} \sum_{p=1}^{N_{\text{part}}} w_p\, p(\mathbf{z}|\mathbf{X}_\mathbf{p})^\alpha, \tag{23}$$

where

$$p(\mathbf{z}) = \sum_{p=1}^{N_{\text{part}}} w_p\, p(\mathbf{z}|\mathbf{X}_p). \tag{24}$$

We wish to choose the sensing action that maximizes the divergence between the current density and the density after a new measurement is acquired. Since we do not know the outcome of a sensing action until after the action is taken, we cannot determine the divergence until after the measurement is made. Therefore, we instead calculate the conditional mean estimate of divergence and use this to schedule the sensors.

Specifically, we calculate the conditional expectation of Eq. (23) given $\mathbf{Z}_t$ for each of the N possible sensing actions and choose the action that maximizes this expected value. Let m refer to the possible sensing action under consideration, including, but not limited to, sensor mode selection and sensor beam positioning.

The expected value of Eq. (23) may be written formally as an integral over all possible outcomes $\mathbf{z}$ when performing sensing action m, i.e.,

$$\langle D_\alpha \rangle_m = \int d\mathbf{z}\, p(\mathbf{z}|\mathbf{Z}_t, m) D_\alpha\big(p(\cdot|\mathbf{Z}_t, \mathbf{z})||p(\cdot|\mathbf{Z}_t)\big). \tag{25}$$

4.2.3. Information based non-myopic sensor management

As discussed earlier, in many situations a non-myopic sensor management strategy provides sensor tasking decisions having better performance than the myopic strategy. In particular, in the setting considered here where targets are "smart" and react to sensing actions, the regret of choosing a poor action persists over time. Therefore, a non-myopic strategy will be far superior to a myopic strategy.

We use batch Q-learning with linear function approximation (see Eq. (10)) to learn a policy which behaves non-myopically and is capable of dynamically adjusting to the environment. In the training process, the immediate reward of an action is computed using the actual gain in information as measured by the Rényi divergence (see Eq. (23)).

5. Simulation results

We consider in this section a model problem in which an airborne platform is to detect and track a set of moving ground targets. The platform has available a multimode sensor able to use an active mode (e.g., radar) or a passive mode (e.g., EO/IR). The sensor is able to quickly steer an antenna so as to focus attention on specific regions of the surveillance area. This is a simple model of a real platform like the USAF JSTARS, which has a 24 ft antenna installed on the underside, is able to scan electronically in azimuth and is able to choose between several modes of operation including moving target indicator and synthetic aperture radar.

In this simulation, targets are characterized by their position in one dimension. Targets are "smart" as they sense when they are under surveillance by an active sensor and react to make future surveillance more difficult. The number and location of the targets is unknown

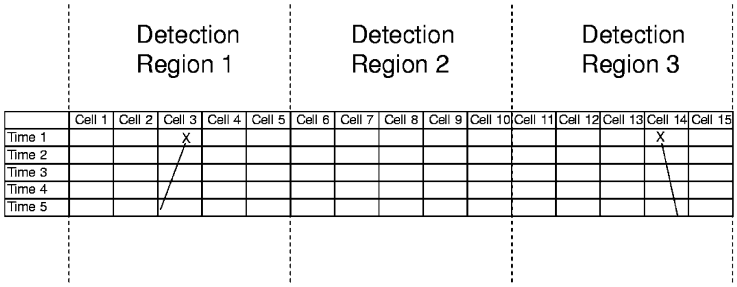

Fig. 2. An illustration of the model problem. The surveillance region is broken into several coarse detection regions, shown as detection regions 1, 2, and 3 above. The detection algorithm schedules the sensor to most quickly determine the presence or absence of targets in each detection region. Upon detecting targets, the tracking algorithm is tipped-off with the regions in which targets exist. The tracking algorithm then determines sensor resource allocations that allows refinement of the initial location and tracking as the targets move through the surveillance area.

initially and our task is to detect and track the targets by selecting the best sensor scheduling policy.

We address the problem following the two step strategy outlined in Section 2. Specifically, we first segment the surveillance area into a set of detection regions where the sensor scheduling strategy of Section 4.1 is used to most quickly determine the presence or absence of a target in each region. Upon determining targets are present, the detection algorithm gives a tip-off to the tracking algorithm of Section 4.2 by providing the information that a target exists and the region in which it exists. The tracking algorithm is then responsible for scheduling the sensor to refine the estimate of target location and track the targets as they move through the surveillance area. The model problem considered here is summarized in Fig. 2.

We proceed in this section to illustrate the two-step detection and tracking algorithm developed using the reinforcement learning (RL) approach. We contrast the performance of this non-myopic strategy to random and myopic strategies.

5.1. Target detection

Each detection region is modelled as taking one of three states: s_1 no target present, s_2 an exposed target is present, and s_3 a camouflaged target is present. There are two hypotheses: H_1 (no target present) and H_2 (a target is present, either exposed or camouflaged). The target can move from state 2 to state 3 if it senses it is being observed. However, it tends to return from state 3 to state 2 if it no longer senses it is being observed, e.g., it may be less effective in state 3.

Intelligence sources provide a prior on the initial state of the target, which constitutes the initial information state of the process $\mathbf{p}_0$. The platform has one of three sensor modes to deploy. Sensor mode i, deployed at time t provides an independent measurement $z_i(t)$. For the simulation considered here, measurements are assumed conditionally Gaussian with the following conditional distributions ($N(\mu, \sigma^2)$ denotes the Gaussian density with mean μ and variance σ^2):

$$
\begin{array}{lll}
z_1(t)|s_1 \sim N(0, 0.45), & z_1(t)|s_2 \sim N(1, 0.45), & z_1(t)|s_3 \sim N(0.4, 0.45), \\
z_2(t)|s_1 \sim N(0, 1), & z_2(t)|s_2 \sim N(1, 1), & z_2(t)|s_3 \sim N(0.4, 1), \\
z_3(t)|s_1 \sim N(0, 0.7), & z_3(t)|s_2 \sim N(1, 0.7), & z_3(t)|s_3 \sim N(0.4, 0.7).
\end{array}
$$

Sensor modes 1 and 3 represent active modes, which can be sensed by the target, and sensor mode 2 represents a lower SNR passive mode which cannot be detected by the target. When the target is in hide mode, it has an incentive to return to the exposed state. To reflect these characteristics the following transition probability matrices were chosen:

$$
\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.2 & 0.1 \\ 0 & 0.8 & 0.9 \end{bmatrix}, \qquad \mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.3 \\ 0 & 0 & 0.7 \end{bmatrix}, \qquad \mathbf{A}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0.05 \\ 0 & 0.9 & 0.95 \end{bmatrix}.
$$

Note that mode 3 is less favorable then sensor mode 1 regardless of system state. It provides less information (lower SNR) on the target, and when it is used there is a higher probability that the target will detect it. It was included in this study to show that the optimal learned policy will indeed never use it.

We assume zero cost for the sensor deployment and a unit reward for a correct decision. Hence, the expected reward (15) becomes $E\{\beta^T \max\{p_T(1), \mathbf{p}_T(2) + \mathbf{p}_T(3)\}\}$, where T is the (random) final decision time. The discount factor β was chosen to be 0.99 to reflect large emphasis on future actions.

Q-learning (Section 3.2) was used to approximate the optimal policy. There is an inherent bias-variance tradeoff (also known as estimation/approximation error tradeoff) in Q-learning. With a fixed training set size, as the number of parameters used to approximate the Q-function decrease, two phenomena occur simultaneously: (1) estimation errors of the parameters decrease, i.e., the variance decreases, and (2) the approximation class becomes smaller and the distance between the true Q-function and its best approximation in the class increases, introducing larger bias. Finding a good approximation class is a problem dependent task which requires experimentation and expert knowledge [19]. We chose the basis functions to be indicator functions of disjoint regions of $\mathcal{S}^3 \times \mathcal{A}$ that correspond to quantization of the simplex $\mathcal{S}^3$ into 55 disjoint regions for each action in $\mathcal{A}$. This was found experimentally to give a reasonable tradeoff between the size of the approximation class and the number of training trajectories needed to achieve good estimation of the parameter θ defined in Eq. (7).

The Q-functions were approximated using 20,000 state-action trajectories in which the initial information state was generated uniformly randomly over $\mathcal{S}^3$. Choosing the size of the training set is one of the important open questions in RL. In practice, one increases the number of samples in the training set until one no longer sees an improvement in the resulting algorithm performance. This procedure was adopted and takes about half an hour using a Pentium M processor running MatLab 6.

The Q-functions associated with each sensor mode are depicted in Fig. 3. The Q-function associated with taking the final decision at state $\mathbf{p}_t$ is known to be $\max\{\mathbf{p}_t(1), \mathbf{p}_t(2) + \mathbf{p}_t(3)\}$ and hence does not need not be estimated. Since $\mathcal{S}^3$ is two-dimensional, all functions are presented over the region $\{[\mathbf{p}(1), \mathbf{p}(2)]: \mathbf{p}(1) \geqslant 0, \mathbf{p}(2) \geqslant 0, \mathbf{p}(1) + \mathbf{p}(2) \leqslant 1\}$ and set to zero outside of this region. The x and y axes correspond to $\mathbf{p}(1)$ and $\mathbf{p}(2)$, respectively. Once the Q-functions were estimated the mapping from $\mathcal{S}^3$ to $\mathcal{A}$ was found

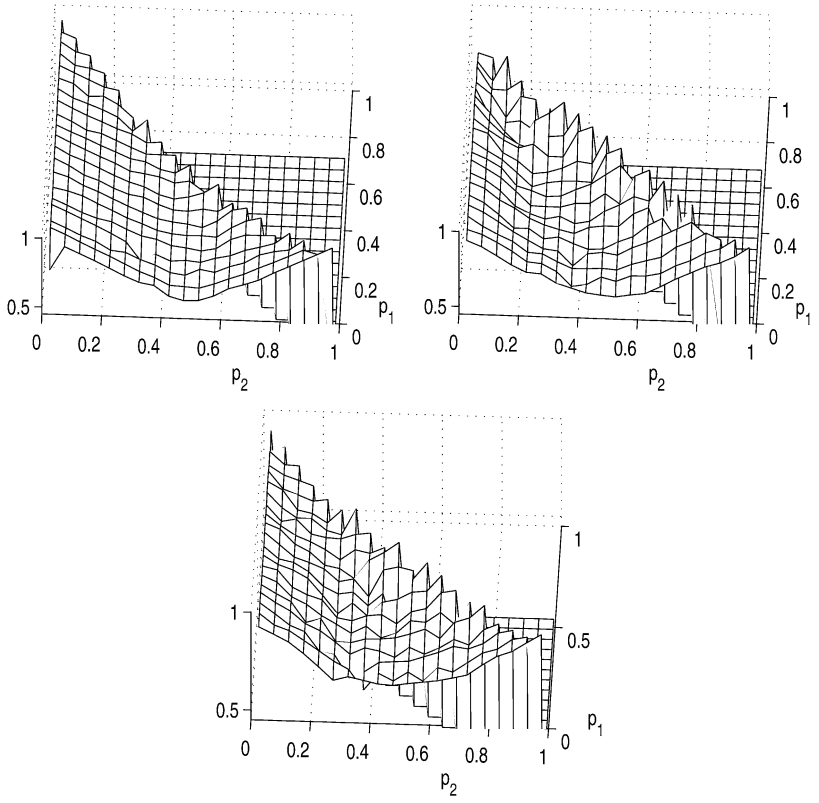

Fig. 3. Q-functions of the three sensor modes over the information space.

according to (11) and is presented in Fig. 4. As expected, sensor 3 is never deployed when using this policy. Furthermore, the policy dictates that the passive sensor is to be used whenever there is a high degree of uncertainty, to make the final decision when either $\mathbf{p}_t(1)$ (decide no target is present) or $\mathbf{p}_t(2) + \mathbf{p}_t(3)$ (decide target present) are close to one, and to use the active sensor only when the final decision is imminent.

As the reward is only collected at the final decision, a myopic strategy is to make an immediate decision based on the prior without taking any measurements. Therefore, the estimated optimal policy is compared to a randomized policy in which actions are chosen uniformly. The improvement in terms of the difference in averaged value, estimated from 2000 Monte Carlo simulations at each information state, is presented in Fig. 5. It is seen that the major difference in value is obtained in the center of the information state space, i.e., when the uncertainty about the system state is maximal.

5.2. Target tracking

We assume for purposes of illustration that the target detection algorithm has correctly detected targets in Regions 1 and 3 (Fig. 2) and passed this information to the target track-

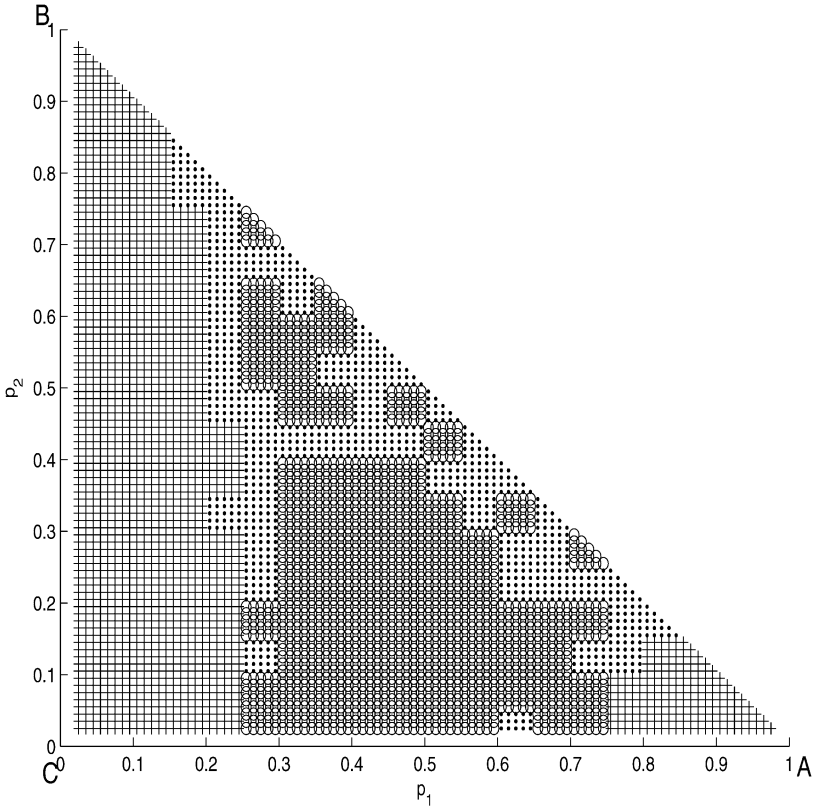

Fig. 4. Sensor allocation map: the dark gray area corresponds to passive mode, the light to active mode and the third area to making the final decision. The points A, B, and C are marked for reference to Fig. 5, as the axis orientation in the two figures is different.

ing algorithm. At each time step, the sensor is able to measure a single cell to determine the presence or absence of targets. Targets can move along a line in a strictly diffusive manner. The sensor can use the active (mode 1) or passive (mode 2) modes described above. Sensor modes are characterized by a detection probability P_{d} and a false alarm probability P_{f}. These probabilities are linked together via SNR by $P_{\mathrm{d}} = P_{\mathrm{f}}^{1/(1+\mathrm{SNR})}$. This model of sensor returns corresponds to thresholding of target return signal in Rayleigh distributed noise as is seen on GMTI radar systems [2]. Note that the sensor characteristics are defined differently than in the detection portion of the algorithm. Unlike the detection regions considered earlier, a sensor cell is now a small area and targets can easily move between cells.

When the target is in visible mode, the active mode works with high detection probability and low false alarm probability, $P_{\mathrm{d}} = 0.9$ and $P_{\mathrm{f}} = 1e-4$ (corresponding to $\mathrm{SNR} = 20$ dB). The passive sensor mode works with detection probability $P_{\mathrm{d}} = 0.5$ and false alarm probability $P_{\mathrm{f}} = 1e-4$ ($\mathrm{SNR} = 10$ dB). When in hide mode, both modes are severely degraded and correspond to a target with $\mathrm{SNR} = 0$ dB.

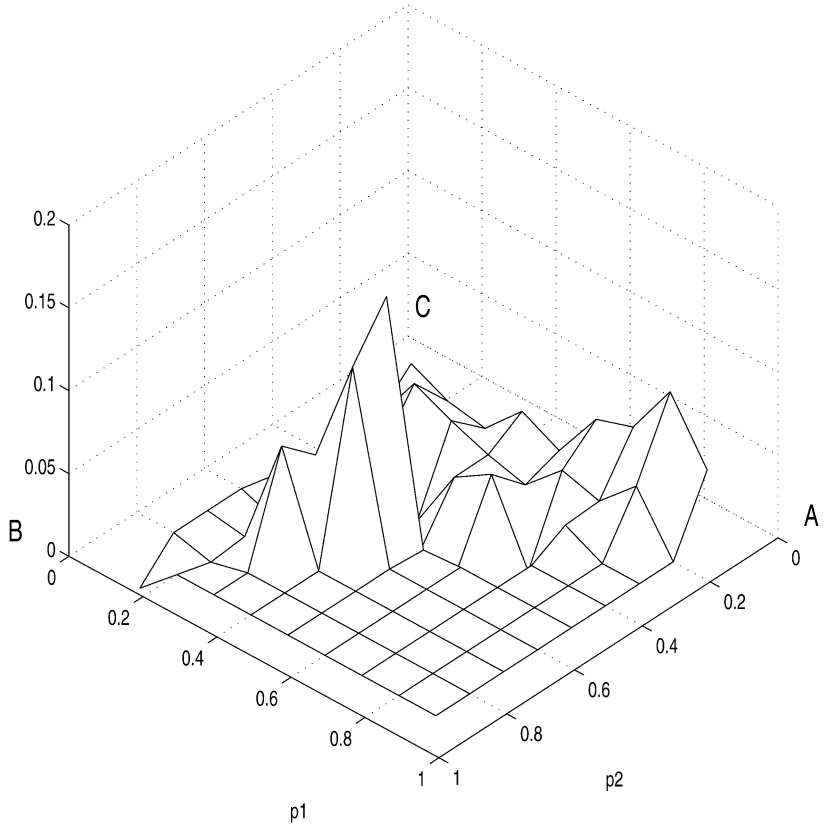

Fig. 5. Improvement over the random allocation policy. Note the relative orientations of the state domain comparison to Fig. 4—the points A, B, and C are marked for reference.

Targets can sense when the active mode is used and move into hide mode to prevent further interrogation. Additionally, targets that have moved into hide mode tend to move back into visible mode when the passive sensor mode is used. The parameters of interest can be summarized by the following transition probabilities when for each of the two sensor modes:

$$\begin{bmatrix} \Pr(\text{visible to visible}) & \Pr(\text{visible to hide}) \\ \Pr(\text{hide to visible}) & \Pr(\text{hide to hide}) \end{bmatrix}.$$

A myopic strategy of sensor management makes tasking decisions based only on the expected immediate reward. Here the myopic strategy will advocate using the active mode at all times as it has the largest expected gain in single step information gain. Depending on the transition probabilities, this may immediately force the targets into hide mode, making them difficult to observe in future time steps. A non-myopic strategy, on the other hand, will take into account the effect of current actions on future information gain and be more prudent in using the active mode.

We illustrate the technique using two simulations with different transition probabilities.

Simulation 1.

$$\text{Transition matrix active sensor mode} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix},$$

$$\text{Transition matrix passive sensor mode} = \begin{bmatrix} 1 & 0 \\ 0.2 & 0.8 \end{bmatrix}.$$

Simulation 2.

$$\text{Transition matrix active sensor mode} = \begin{bmatrix} 0.1 & 0.9 \\ 0 & 1 \end{bmatrix},$$

$$\text{Transition matrix passive sensor mode} = \begin{bmatrix} 1 & 0 \\ 0.33 & 0.67 \end{bmatrix}.$$

In simulation 1, the target always moves into hide when the active mode is used and moves from hide to visible with probability 0.2 when the active mode is used. In simulation 2, the target has a 10% chance of remaining in visible mode even if the active mode is used, and is more likely to move back into visible mode when the passive mode is used.

We trained a Q-function as discussed in Section 2. Episodes were generated with random sensor allocations using the models of target behavior. The initial position of the targets and realization of the diffusive motion were chosen randomly for each training episode. The Q-function was trained using a linear function approximation on 100,000 training episodes in batch fashion. Table 1 gives empirical results for how performance of the algorithm improves as the number of training episodes is increased, showing 100,000 is a good stopping point. The algorithm was tested on 1000 example episodes where the initial position and realization of the diffusive motion of the targets was chosen randomly for each testing episode. The Q-function learned during the training episode was used to schedule the sensor by selecting mode and pointing direction.

In Figs. 6–8, we present results of Q-learning performance on the tracking stage. We compare performance to (a) a random strategy, (b) a myopic strategy, (c) a random strategy that only uses the passive mode, and (d) a myopic strategy that only uses the passive mode. The Q-learning strategy performs as well or better than the best of the four competing strategies in both cases.

Table 1
Performance versus training examples

Training episodes	Tracking error (m)
200	1.6975
1000	1.5996
5000	1.5603
10,000	1.5126
50,000	1.4758
100,000	1.4071
200,000	1.4103

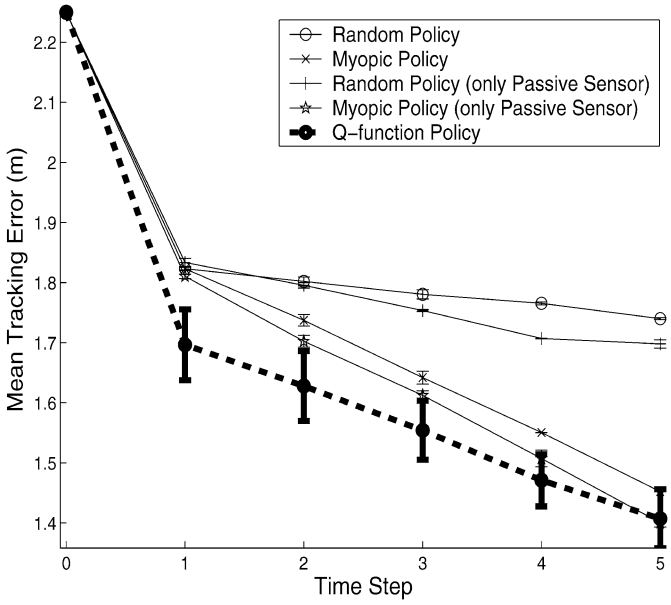

Fig. 6. Target tracking performance, in terms of average tracking error for simulation 1. Included are a random strategy, a myopic strategy, a random strategy that uses only the passive mode, a myopic strategy that uses only the passive mode, and the Q-learning strategy.

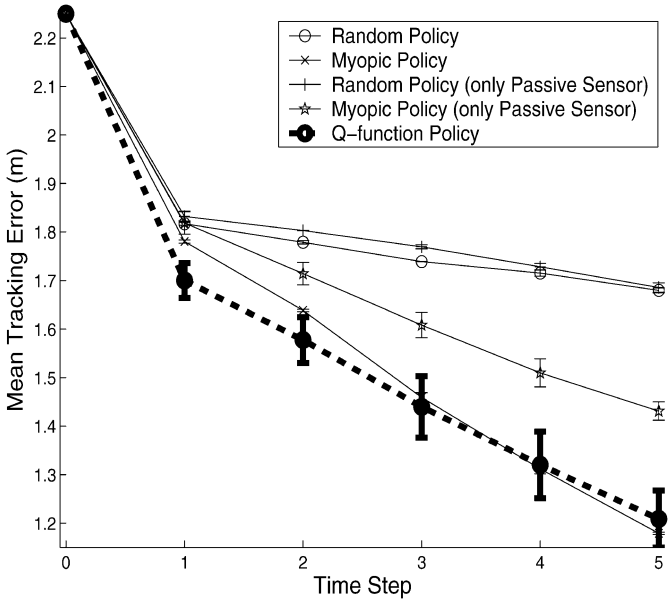

Fig. 7. Target tracking performance, in terms of average tracking error for simulation 2. Included are a random strategy, a myopic strategy, a random strategy that uses only the passive sensor, a myopic strategy that uses only the passive mode, and the Q-learning strategy.

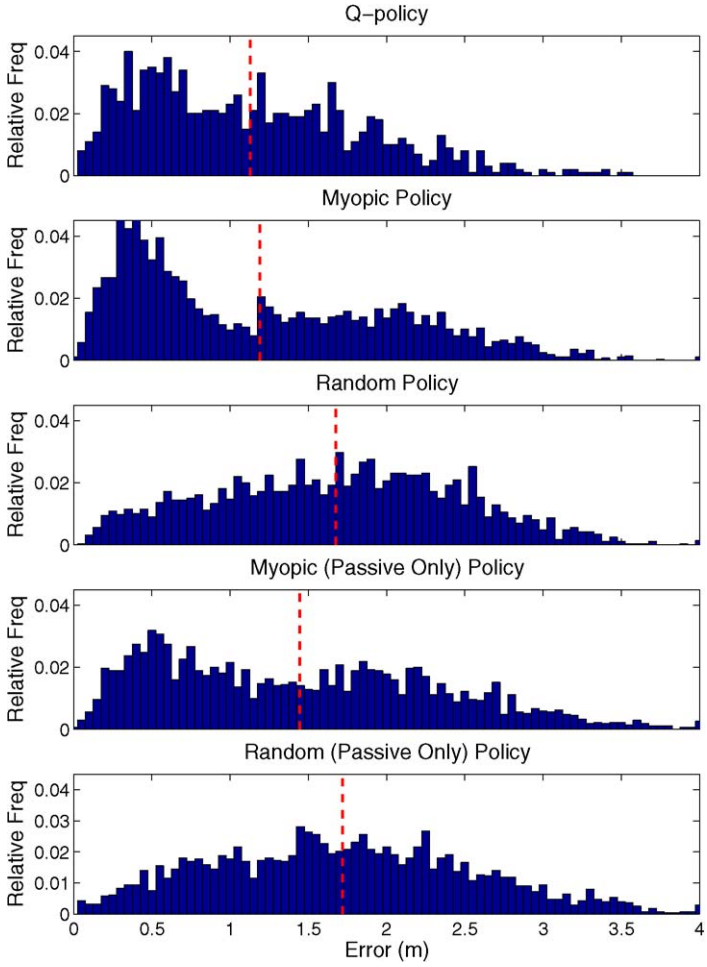

Fig. 8. Histograms showing relative frequency of magnitude tracking errors at time 5 for simulation 1 when testing the five polices considered here. The mean performance of each policy is given by the dashed line. Note that the performance of the tracker is bimodal—either the tracker finds the target (error $\sim$ 1 m or less) or it does not (error $>$ 1 m).

6. Conclusion

In this paper, we have investigated the problem of sensor scheduling for detection and tracking of smart moving ground targets from an airborne sensor. Since the targets of interest are able to detect and respond to certain sensing actions, it is mandatory that the long term ramifications be taken into account when choosing current sensing actions. This necessity for non-myopic sensor scheduling leads to a very computationally challenging problem.

We have addressed this numerical challenge with a two stage approach. The surveillance area is first partitioned into a set of detection regions and a detection algorithm determines

the presence or absence of a target in each region. Upon detection, a tracking algorithm is used to finely geolocate and track targets as they move through the region.

The sensor scheduling algorithm for both of the stages stages was developed with a reinforcement learning (RL) approach. Out method relies on a set of training data used in batch to learn a good sensor management policy. We showed through a series of experiments that the RL approach allows accommodation of the desire to perform quick detection and performs as well or better than other simple strategies in the tracking problem. An alternative approach for future research is to learn a good policy on-line while another policy is being executed.

Acknowledgments

This work was supported by the USAF Contract No. F33615-02-C-1199, AFRL contract SPO900-96-D-0080, and ARO-DARPA MURI Grant DAAD19-02-1-0262. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

References

[1] G. Apostolikas, S. Tzafestas, Improved Q (MDP) policy partially observable Markov decision processes in large domains: embedding exploration dynamics, Intell. Automat. Soft Comput. 10 (3) (2004) 209–220.
[2] Y. Bar-Shalom, X. Li, Estimation and Tracking: Principles, Techniques, and Software, Artech House, 1993.
[3] D.P. Bertsekas, Dynamic Programming: Deterministic and Stochastic Models, Prentice–Hall, 1987.
[4] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, 1996.
[5] D.P. Bertsekas, D. Castanon, Rollout algorithms for stochastic scheduling problems, J. Heurist. 5 (1999) 89–108.
[6] A. Doucet, N. de Freitas, N. Gordon, Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York, 2001.
[7] A.O. Hero, B. Ma, O. Michel, J. Gorman, Applications of entropic spanning graphs, IEEE Signal Process. Mag. Special Issue Math. Imag. 19 (5) (2002) 85–95.
[8] K.J. Hintz, E.S. McVey, Multi-process constrained estimation, IEEE Trans. Man, Syst. Cybernet. 21 (1) (1991) 434–442.
[9] C. Kreucher, K. Kastella, A.O. Hero, Tracking multiple targets using a particle filter representation of the joint multitarget probability density, in: SPIE International Symposium on Optical Science and Technology, San Diego, CA, August 2003.
[10] C. Kreucher, K. Kastella, A.O. Hero, Information-based sensor management for multitarget tracking, in: SPIE International Symposium on Optical Science and Technology, San Diego, CA, August 2003.
[11] V. Krishnamurthy, Algorithms for optimal scheduling and management of hidden Markov model sensors, IEEE Trans. Signal Process. 50 (6) (2002) 1382–1397.
[12] V. Krishnamurthy, D. Evans, Hidden Markov model multiarm bandits: A methodology for beam scheduling in multitarget tracking, IEEE Trans. Signal Process. 49 (2001) 2893–2908.
[13] R. Mahler, Global optimal sensor allocation, in: Proceedings of the Ninth National Symposium on Sensor Fusion, vol. I, Naval Postgraduate School, Monterey, CA, 12–14 March, 1996, pp. 347–366.
[14] M. Mundhenk, J. Goldsmith, C. Lusena, E. Allender, Complexity of finite-horizon Markov decision process problems, J. ACM 47 (4) (2000) 681–720.
[15] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, 1994.
[16] A. Rényi, On measures of entropy and information, in: Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics, and Probability, vol. 1, 1961, pp. 547–561.

[17] W. Schmaedeke, K. Kastella, Event-averaged maximum likelihood estimation and information-based sensor management, in: Proceedings of SPIE, vol. 2232, Orlando, FL, June 1994, pp. 91–96.
[18] R.S. Sutton, A.G. Barto, Reinforcement Learning, MIT Press, 1998.
[19] B. Van Roy, Neuro-dynamic programming: overview and recent trends, in: Handbook of Markov Decision Processes: Methods and Applications, 2001.
[20] C. Watkins, Learning from delayed rewards, Thesis, University of Cambridge, England, 1989.
[21] F. Zhao, J. Shin, J. Reich, Information-driven dynamic sensor collaboration, IEEE Signal Process. Mag. (2002) 61–72.