Partially Observable Markov Decision Process Approximations for Adaptive Sensing

Edwin K. P. Chong, Christopher M. Kreucher, and Alfred O. Hero III

Abstract

Adaptive sensing involves actively managing sensor resources to achieve a sensing task, such as object detection, classification, and tracking, and represents a promising direction for new applications of discrete event system methods. We describe an approach to adaptive sensing based on approximately solving a partially observable Markov decision process (POMDP) formulation of the problem. Such approximations are necessary because of the very large state space involved in practical adaptive sensing problems, precluding exact computation of optimal solutions. We review the theory of POMDPs and show how the theory applies to adaptive sensing problems. We then describe a variety of approximation methods, with examples to illustrate their application in adaptive sensing. The examples also demonstrate the gains that are possible from nonmyopic methods relative to myopic methods, and highlight some insights into the dependence of such gains on the sensing resources and environment.

I. Introduction

A. What is Adaptive Sensing?

In its broadest sense, *adaptive sensing* has to do with actively managing sensor resources to achieve a sensing task. As an example, suppose our goal is to determine the presence or absence of an object, and we have at our disposal a single sensor that can interrogate the scene with any one of K waveforms. Depending on which waveform is used to irradiate the scene, the response may vary greatly. After each measurement, we can decide whether to continue taking measurements using that waveform, change waveforms and take further measurements, or stop and declare whether or not the object is present. In adaptive sensing, this decision making is allowed to take advantage of the knowledge gained from the measurements so far. In this sense, the act of sensing "adapts" to what we know so far. What guides this adaptation is a performance objective that is determined beforehand—in our example above, this might be the average number of interrogations needed so that we can declare the presence or absence of the object with a confidence that exceeds some threshold (say, 90%).

Adaptive sensing problems arise in a variety of application areas, and represent a promising direction for new applications of discrete event system methods. Here, we outline only a few.

Medical diagnostics. Perhaps the most familiar example of adaptive sensing takes place between a doctor and a patient. The task here is to diagnose an illness from a set of symptoms, using a variety of medical tests at the doctor's disposal. These include physical examinations, blood tests, radiographs (X-ray images), computerized tomography (CT) scans, and magnetic resonance imaging (MRI). Doctors use results from tests so far to determine what test to perform next, if any, before making a diagnosis.

Nondestructive testing. In nondestructive testing, the goal is to use noninvasive methods to determine the integrity of a material or to measure some characteristic of an object. A wide variety of methods are used in nondestructive testing, ranging from optical to microwave to acoustic. Often, several methods must be used before a determination can be made. The test results obtained so far inform what method to use next (including what waveform to select), thus giving rise to an instance of adaptive sensing.

Edwin K. P. Chong is with Colorado State University. Email: edwin.chong@colostate.edu
Christopher M. Kreucher is with Integrity Applications Incorporated in Ann Arbor, MI. Email: ckreuche@umich.edu
Alfred O. Hero III is with the University of Michigan. Email: hero@umich.edu

This material is based in part upon work supported by the Air Force Office of Scientific Research under Award FA9550-06-1-0324 and by DARPA under Award FA8750-05-2-0285. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Air Force or of DARPA. Approved for Public Release, Distribution Unlimited.

Sensor scheduling for target detection, identification, and tracking. Imagine a group of airborne sensors—say, radars on unmanned aerial vehicles (UAVs)—with the task of detecting, identifying, and tracking one or more targets on the ground. For a variety of reasons, we can use at most one sensor at any given time. These reasons include limitations in communication resources needed to transmit data from the sensors, and the desire to minimize radar usage to maintain covertness. The selection of which sensor to use over time is called sensor scheduling, and is an adaptive sensing problem.

Waveform selection for radar imaging. Radar systems have become sufficiently agile that they can be programmed to use waveform pulses from a library of waveforms. The response of a target in the scene can vary greatly depending on what waveform is used to radiate the area due to intrapulse characteristics (e.g., frequency and bandwidth) or interpulse characteristics (e.g., pulse repetition interval). The main issue in the operation of such agile radar systems is the selection of waveforms to use in a particular scenario. If past responses can be used to guide the selection of waveforms, then this issue is an instance of adaptive sensing.

Laser pulse shaping. Similar to the last example, optical waveforms can also be designed to generate a variety of responses, only at much smaller wavelengths. By carefully tailoring the shape of intense light pulses, the interaction of light with even a single atom can be controlled [2]. The possibility of such controlled interactions of light with atoms has many promising applications. As in the previous example, these applications give rise to adaptive sensing problems.

B. Nonmyopic Adaptive Sensing

In our view, adaptive sensing is fundamentally a *resource management* problem, in the sense that the main task is to make decisions over time on the use of sensor resources to maximize sensing performance. It is informative to distinguish between *myopic* and *nonmyopic* (also known as *dynamic* or *multistage*) resource management, a topic of much current interest (see, e.g., [32], [22], [5], [23], [27], [40], [41], [25]). In myopic resource management, the objective is to optimize performance on a per-decision basis. For example, consider the problem of *sensor scheduling* for tracking a single target, where the problem is to select, at each decision epoch, a single sensor to activate. An example sensor-scheduling scheme is *closest point of approach*, which selects the sensor that is perceived to be the closest to the target. Another (more sophisticated) example is the method described in [34], where the authors present a sensor scheduling method using alpha-divergence (or Rényi divergence) measures. Their approach is to make the decision that maximizes the expected information gain (in terms of the alpha-divergence).

Myopic adaptive sensing may not be ideal when the performance is measured over a horizon of time. In such situations, we need to consider schemes that trade off short-term for long-term performance. We call such schemes *nonmyopic*. Several factors motivate the consideration of nonmyopic schemes, easily illustrated in the context of sensor scheduling for target tracking:

Heterogeneous sensors. If we have sensors with different locations, waveform characteristics, usage costs, and/or lifetimes, the decision of whether or not to use a sensor, and with what waveform, should consider the overall performance, not whether or not its use maximizes the current performance.

Sensor motion. The future location of a sensor affects how we should act now. To optimize a long-term performance measure, we need to be opportunistic in our choice of sensor decisions.

Target motion. If a target is moving, there is potential benefit in sensing the target before it becomes unresolvable (e.g., too close to other targets or to clutter, or shadowed by large objects). In some scenarios, we may need to identify multiple targets before they cross, to aid in data association.

Environmental variation. Time-varying weather patterns affect target visibility in a way that potentially benefits from nonmyopic decision making. In particular, by exploiting models of target visibility maps, we can achieve improved sensing performance by careful selection of waveforms and beam directions over time. We show an example along these lines in Section VIII.

The main focus of this paper is on nonmyopic adaptive sensing. The basic methodology presented here consists of two steps:

1) Formulating the adaptive sensing problem as a partially observable Markov decision process (POMDP); and
2) Applying an approximation to the optimal policy for the POMDP, because computing the exact solution is intractable.

Our contribution is severalfold. First, we show in detail how to formulate adaptive sensing problems in the framework of POMDPs. Second, we survey a number of approximation methods for such POMDPs. Our treatment of these methods includes their underlying foundations and practical considerations in their implementation. Third, we illustrate the performance gains that can be achieved via examples. Fourth, in our illustrative examples, we highlight some insights that are relevant to adaptive sensing problems: (1) with very limited sensing resources, nonmyopic sensor and waveform scheduling can significantly outperform myopic methods with only moderate increase in computational complexity; and (2) as the number of available resources increases, the nonmyopic advantage decreases.

Significant interest in nonmyopic adaptive sensing has arisen in the recent robotics literature. For example, the recent book by Thrun, Burgard, and Fox [54] describes examples of such approaches, under the rubric of *probabilistic robotics*. Our paper aims to address increasing interest in the subject in the signal processing area as well. Our aim is to provide an accessible and expository treatment of the subject, introducing a class of new solutions to what is increasingly recognized to be an important new problem.

C. Paper Organization

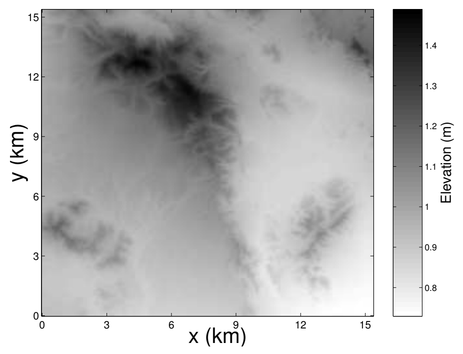
This paper is organized as follows. In Section II, we give a concrete motivating example that advocates the use of nonmyopic methods. We then describe, in Section III, a formulation of the adaptive sensing problem as a partially observable Markov decision process (POMDP). We provide three examples to illustrate how to formulate adaptive sensing problems in the POMDP framework. Next, in Section IV, we review the basic principles behind Q-value approximation, the key idea in our approach. Then, in Section V, we illustrate the basic lookahead control framework and describe the constituent components. In Section VI, we describe a host of Q-value approximation methods. Among others, this section includes descriptions of Monte Carlo sampling methods, heuristic approximations, rollout methods, and the traditional reinforcement learning approach. In Sections VII and VIII, we provide simulation results on model problems that illustrate several of the approximate nonmyopic methods described in this paper. We conclude in Section IX with some summary remarks.

In addition to providing an expository treatment on the application of POMDPs to the adaptive sensing problem, this paper includes several new and important contributions. First, we introduce a model problem that includes time-varying intervisibility which has all of the desirable properties to completely explore the trade between nonmyopic and myopic scheduling. Second, we introduce several potentially tractable and general numerical methods for generating approximately optimal nonmyopic policies, and show explicitly how they relate to the optimal solution. These include belief-state simplification, completely observable rollout, and reward surrogation, as well as a heuristic based on an information theoretic approximation to the value-to-go function which is applicable in a broad array of scenarios (these contributions have never appeared in journal publications). Finally, these new techniques are compared on a model problem, followed by an in-depth illustration of the value of nonmyopic scheduling on the model problem.

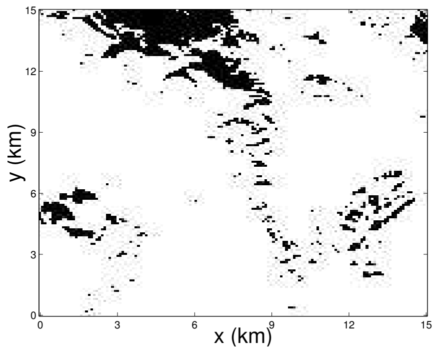
II. Motivating Example

We now present a concrete motivating example that will be used to explain and justify the heuristics and approximations used in this paper. This example involves a remote sensing application where the goal is to learn the contents of a surveillance region via repeated interrogation. (See [24] for a more complete exposition of adaptive sensing applied to such problems.)

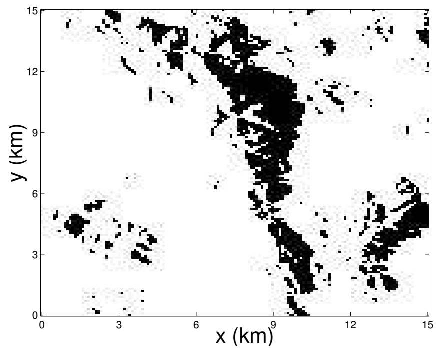
Consider a single airborne sensor which is able to image a portion of a ground surveillance region to determine the presence or absence of moving ground targets. At each time epoch, the sensor is able to direct an electrically scanned array so as to interrogate a small area on the ground. Each interrogation



(a) Elevation map of the surveillance region



(b) Visibility mask for a sensor south of the region



(c) Visibility mask for a sensor west of the region

Fig. 1. Top: A digital terrain elevation map for a surveillance region, indicating the height of the terrain in the region. Bottom: Visibility masks for a sensor positioned to the south and to the west, respectively, of the surveillance region. We show binary visibility masks (nonvisible areas are black and visible areas are white). In general, visibility may be between 0 and 1 indicating areas of reduced visibility, e.g., regions that are partially obscured by foliage.

yields some (imperfect) information about the small area. The objective is to choose the sequence of pointing directions that lead to the best ability to estimate the entire contents of the surveillance region.

Further complicating matters is the fact that at each time epoch the sensor position causes portions of the ground to be unobservable due to the terrain elevation between the sensor and the ground. Given its position and the terrain elevation, the sensor can compute a visibility mask which determines how well a particular spot on the ground can be seen. As an example, in Figure 1 we give binary visibility masks computed from a sensor positioned (a) south and (b) to the west of the topologically nonhomogeneous surveillance region (these plots come from real digital terrain elevation maps). As can be seen from the figures, sensor position causes “shadowing” of certain regions. These regions, if measured, would provide no information to the sensor. A similar target masking effect occurs with atmospheric propagation attenuation from disturbances such as fog, rain, sleet, or dust, as illustrated in Section VIII. This example illustrates a situation where nonmyopic adaptive sensing is highly beneficial. Using a known sensor trajectory and known topological map, the sensor can predict locations that will be obscured in the future. This information can be used to prioritize resources so that they are used on targets that are predicted to become obscured in the future. Extra sensor dwells immediately before obscuration (at the expense of not interrogating other targets) will sharpen the estimate of target location. This sharpened estimate will allow better prediction of where and when the target will emerge from the obscured area. This is illustrated graphically with a six time-step vignette in Figure 2.

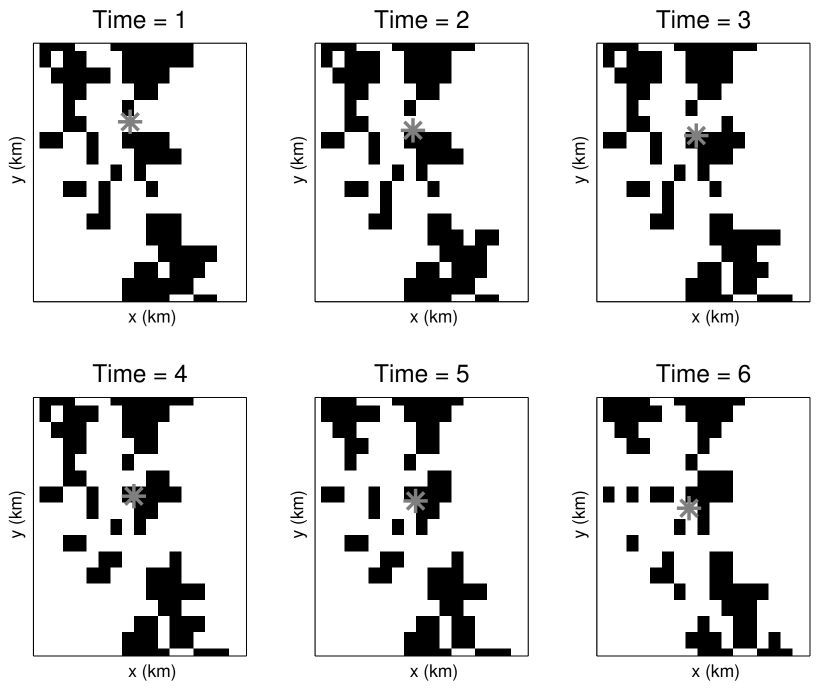


Fig. 2. A six time step vignette where a target moves through an obscured area. Other targets are present elsewhere in the surveillance region. The target is depicted by an asterisk. Areas obscured to the sensor are black and areas that are visible are white. Extra dwells just before becoming obscured (time = 1) aid in localization after the target emerges (time = 6).

III. Formulating Adaptive Sensing Problems

A. Partially Observable Markov Decision Processes

An adaptive sensing problem can be posed formally as a *partially observable Markov decision process* (POMDP). Before discussing exactly how this is done, we first need to introduce POMDPs. Our level of treatment will not be as formal and rigorous as one would expect from a fullblown course on this topic. Instead, we seek to describe POMDPs in sufficient detail to allow the reader to see how an adaptive sensing problem can be posed as a POMDP, and to explore methods to approximate optimal solutions. Our exposition assumes knowledge of probability, stochastic processes, and optimization. In particular, we assume some knowledge of Markov processes, including Markov decision processes, a model that should be familiar to the discrete event system community. For completeness, we will introduce POMDPs in sufficient detail to allow the reader to see how an adaptive sensing problem can be posed as a POMDP, and to explore methods to approximate optimal solutions. For a full treatment of POMDPs and related background, see [6].

A POMDP is specified by the following ingredients:

- A set of states (the state space) and a distribution specifying the random initial state.
- A set of possible actions (the action space).
- A state-transition law specifying the next-state distribution given an action taken at a current state.
- A reward function specifying the reward (real number) received given an action taken at a state.
- A set of possible observations (the observation space).
- An observation law specifying the distribution of observations given an action taken at a state.

A POMDP is a controlled dynamical process in discrete time. The process begins at time $k = 0$ with a (random) initial state. At this state, we perform an action and receive a reward, which depends on the action and the state. At the same time, we receive an observation, which again depends on the action and the state. The state then transitions to some random next state, whose distribution is specified by the state-transition law. The process then repeats in the same way—at each time, the process is at some state, and the action taken at that state determines the reward, observation, and next state. As a result, the state evolves randomly over time in response to actions, generating observations along the way.

We have not said anything so far about the finiteness of the state space or the sets of actions and observations. The advantage to leaving this issue open is that it frees us to construct models in the most natural way. Of course, if we are to represent any such model in a computer, we can only do so in a finite way (though the finite numbers that can be represented in a computer are typically sufficiently large to meet practical needs). For example, if we model the motion of a target on the ground in terms of its Cartesian coordinates, we can deal with this model in a computer only in a finite sense—specifically, there are only a finite number of possible locations that can be captured on a standard digital computer. Moreover, the theory of POMDPs becomes much more technically involved if we are to deal rigorously with infinite sets. For the sake of technical formality, we will assume henceforth that the state space, the action space, and the observation space are all finite (though not necessarily "small"—we stress that this assumption is merely for technical reasons). However, when thinking about models, we will not explicitly restrict ourselves to finite sets. For example, it is convenient to use a motion model for targets in which we view the Cartesian coordinates as real numbers. There is no harm in this dichotomous approach as long as we understand that ultimately we are computing only with finite sets.

B. Belief State

As a POMDP evolves over time, we do not have direct access to the states that occur. Instead, all we have are the observations generated over time, providing us with clues of the actual underlying states (hence the term *partially observable*). These observations might, in some cases, allow us to infer exactly what states actually occurred. However, in general, there will be some uncertainty in our knowledge of the states that actually occurred. This uncertainty is represented by the *belief state* (or *information* *state*), which is the *a posteriori* (or *posterior*) distribution of the underlying state given the history of observations.

Let $\mathcal{X}$ denote the state space (the set of all possible states in our POMDP), and let $\mathcal{B}$ be the set of distributions over $\mathcal{X}$. Then a belief state is simply an element of $\mathcal{B}$. Just as the underlying state changes over time, the belief state also changes over time. At time $k = 0$, the (initial) belief state is equal to the given initial state distribution. Then, once an action is taken and an observation is received, the belief state changes to a new belief state, in a way that depends on the observation received and the state-transition and observation laws. This change in the belief state can be computed explicitly using Bayes' rule.

To elaborate, suppose that the current time is k, and the current belief state is $b_k \in \mathcal{B}$. Note that b_k is a probability distribution over $\mathcal{X}$—we use the notation $b_k(x)$ for the probability that b_k assigns to state $x \in \mathcal{X}$. Let $\mathcal{A}$ represent the action space. Suppose that at time k we take action $a_k \in \mathcal{A}$ and, as a result, we receive observation y_k. Denote the state-transition law by P_{trans}, so that the probability of transitioning to state x' given that action a is taken at state x is $P_{\text{trans}}(x'|x, a)$. Similarly, denote the observation law by P_{obs}, so that the probability of receiving observation y given that action a is taken at state x is $P_{\text{obs}}(y|x, a)$. Then, the next belief state given action a_k is computed using the following two-step update procedure:

1. Compute the "updated" belief state $\hat{b}_k$ based on the observation y_k of the state x_k at time k, using Bayes' rule:
$$\hat{b}_k(x) = \frac{P_{\text{obs}}(y_k|x, a_k)b_k(x)}{\sum_{s\in\mathcal{X}} P_{\text{obs}}(y_k|s, a_k)b_k(s)}, \quad x \in \mathcal{X}.$$
2. Compute the belief state b_{k+1} using the state-transition law:
$$b_{k+1}(x) = \sum_{s\in\mathcal{X}} \hat{b}_k(s)P_{\text{trans}}(x|s, a_k), \quad x \in \mathcal{X}.$$

This two-step procedure is commonly realized in terms of a Kalman filter or a particle filter [46].

It is useful to think of a POMDP as a random process of evolving belief states. Just as the underlying state transitions to some random new state with the performance of an action at each time, the belief state also transitions to some random new belief state. So the belief state process also has some "belief-state-transition" law associated with it, which depends intimately on the underlying state-transition and the observation laws. But, unlike the underlying state, the belief state is fully accessible.

Indeed, any POMDP may be viewed as a *fully observable* Markov decision process (MDP) with state space $\mathcal{B}$, called the *belief-state MDP* or *information-state MDP* (see [6]). To complete the description of this MDP, we will show how to write its reward function, which specifies the reward received when action a is taken at belief-state b. Suppose $b \in \mathcal{B}$ is some belief state and a is an action. Let $R(x, a)$ be the reward received if action a is taken at underlying state x. Then let $r(b, a) = \sum_{x\in\mathcal{X}} b(x)R(x, a)$ be the expected reward with respect to belief-state b, given action a. This reward $r(b, a)$ then represents the reward function of the belief-state MDP.

C. Optimization Objective

Given a POMDP, our goal is to select actions over time to maximize the expected cumulative reward (we take expectation here because the cumulative reward is a random variable). To be specific, suppose we are interested in the expected cumulative reward over a time horizon of length H: $k = 0, 1, \ldots, H-1$. Let x_k and a_k be the state and action at time k, and let $R(x_k, a_k)$ be the resulting reward received. Then, the cumulative reward over horizon H is given by

$$V_H = \mathrm{E}\left[\sum_{k=0}^{H-1} R(x_k, a_k)\right],$$

where E represents expectation. It is important to realize that this expectation is with respect to $x_0, x_1, \ldots$; i.e., the random initial state and all the subsequent states in the evolution of the process, given the actions $a_0, a_1, a_2, \ldots$ taken over time. The goal is to pick these actions so that the objective function is maximized.

We have assumed without loss of generality that the reward is a function only of the current state and the action. Indeed, suppose we write the reward such that it depends on the current state, the next state, and the action. We can then take the conditional mean of this reward with respect to the next state, given the current state and action (the conditional distribution of the next state is given by the state-transition law). Because the overall objective function involves expectation, replacing the original reward with its conditional mean in the way described above results in no loss of generality. Finally, notice that the conditional mean of the original reward is a function of the current state and the action, but not the next state.

Note that we can also represent the objective function in terms of r (the reward function of the belief-state MDP) instead of R:

$$V_H(b_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} r(b_k, a_k) \middle| b_0\right].$$

where $\mathrm{E}[\cdot|b_0]$ represents conditional expectation given b_0. The expectation now is with respect to $b_0, b_1, \ldots$; i.e., the initial belief state and all the subsequent belief states in the evolution of the process. We leave it to the reader to verify this expression involving belief states indeed gives rise to the same objective function value as the earlier expression involving states. In Section IV we will discuss an equation, due to Bellman, that characterizes this conditional form of the objective function.

It is often the case that the horizon H is very large. In such cases, for technical reasons relevant to the analysis of POMDPs, the objective function is often expressed as a limit. A sensible limiting objective function is the *infinite-horizon* (or *long-term*) *average* reward:

$$\lim_{H\to\infty} \mathrm{E}\left[\frac{1}{H}\sum_{k=0}^{H-1} R(x_k, a_k)\right].$$

Another common limiting objective function is the *infinite-horizon cumulative discounted* reward:

$$\lim_{H\to\infty} \mathrm{E}\left[\sum_{k=0}^{H-1} \gamma^k R(x_k, a_k)\right],$$

where $\gamma \in (0,1)$ is called the *discount factor*. In this paper, our focus is not on analytical approaches to solving POMDPs. Therefore, even when dealing with large horizons, we will not be concerned with the technical considerations involved in taking the kinds of limits in the above infinite-horizon objective functions [6]. Instead, we will often imagine that H is very large but still use the nonlimiting form.

D. Optimal Policy

In general, the action chosen at each time should be allowed to depend on the entire history up to that time (i.e., the action at time k is a random variable that is a function of all observable quantities up to time k). However, it turns out that if an optimal choice of such a sequence of actions exists, then there is an optimal choice of actions that depends only on "belief-state feedback" (see [52] and references therein for the origins of this result). In other words, it suffices for the action at time k to depend only on the belief-state b_k at time k. So what we seek is, at each time k, a mapping $\pi_k^* : \mathcal{B} \to \mathcal{A}$ such that if we perform action $a_k = \pi_k^*(b_k)$, then the resulting objective function is maximized. As usual, we call such a mapping a *policy*. So, what we seek is an *optimal policy*.

E. POMDPs for Adaptive Sensing

POMDPs form a very general framework based on which many different stochastic control problems can be posed. Thus, it is no surprise that adaptive sensing problems can be posed as POMDPs.

To formulate an adaptive sensing problem as a POMDP, we need to specify the POMDP ingredients in terms of the given adaptive sensing problem. This specification is problem specific. To show the reader how this is done, here we provide some examples of what aspects of adaptive sensing problems influence how the POMDP ingredients are specified. As a further illustration, in the next three sections we specify POMDP models for three example problems, including the motivating example in Section II and the simulations.

States. The POMDP state represents those features in the system (directly observable or not) that possibly evolve over time. Typically, the state is composed of several parts. These include target positions and velocities, sensor modes of operation, sensor parameter settings, battery status, data quality, which sensors are active, states that are internal to tracking algorithms, the position and connectivity of sensors, and communication resource allocation.

Actions. To specify the actions, we need to identify all the controllable aspects of the sensing system (those aspects that we wish to control over time in our adaptive sensing problem). These include sensor mode switching (e.g., waveform selection or carrier frequencies), pointing directions, sensor tunable parameters, sensor activation status (on/off), sensor position changes, and communication resource reallocation.

State-transition law. The state-transition law is derived from models representing how states change over time. Some of these changes are autonomous, while some are in response to actions. Examples of such changes include target motion, which sensors were most recently activated, changes in sensor parameter settings, sensor failures over time, battery status changes based on usage, and changes in the position and connectivity of sensors.

Reward function. To determine the reward function, we need to first decide on our overall objective function. To be amenable to POMDP methods, this objective function must be of the form shown before, namely the mean sum of per-time-step rewards. Writing the objective function this way automatically specifies the reward function. For example, if the objective function is the mean cumulative tracking error, then the reward function simply maps the state at each time to the mean tracking error at that time.

Observations. The observation at each time represents those features of the system that depend on the state and are accessible to the controlling agent (i.e., can be used to inform control decisions). These include sensor outputs (e.g., measurements of target locations and velocities), and those parts of state that are directly observable (e.g., battery status), including prior actions.

Observation law. The observation law is derived from models of how the observations are related to the underlying states. In particular, we will need to use models of sensors (i.e., the relationship between the sensor outputs and the quantities being measured), and also models of the sensor network configuration.

In the next three sections, we provide examples to illustrate how to formulate adaptive sensing problems as POMDPs. In the next section, we show how to formulate an adaptive *classification* problem as a POMDP (with detection problems being special cases). Then, in the section that follows, we show how to formulate an adaptive *tracking* problem as a POMDP. Finally, we consider the airborne sensing problem in Section II and describe a POMDP formulation for it. (which also applies to the simulation example in Section VII).

F. POMDP for an Adaptive Classification Problem

We now consider a simple classification problem and show how the POMDP framework can be used to formulate this problem. In particular, we will give specific forms for each of the ingredients described in Section III-E. This simple classification problem statement can be used to model problems such as medical diagnostics, nondestructive testing, and sensor scheduling for target detection.

Our problem in illustrated in Figure 3. Suppose an object belongs to a particular unknown class c, taking values in a set $\mathcal{C}$ of possible classes. We can take measurements on the object that provide us with information from which we will infer the unknown class. These measurements come from a "controlled sensor" at our disposal, which we can use at will. Each time we use the sensor, we first have to choose a control $u \in \mathcal{U}$. For each chosen control u, we get a measurement whose distribution depends on c and u. Call this distribution $P_{\text{sensor}}(\cdot|c, u)$ (repeated uses of the sensor generate independent measurements). Each time we apply control u, we incur a cost of $\kappa(u)$ (i.e., the cost of using the controlled sensor depends on

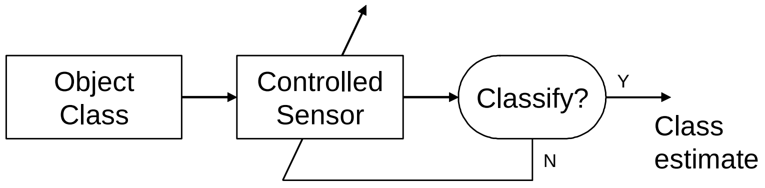


Fig. 3. An adaptive classification system.

the control applied). The controlled sensor may represent a particular measurement instrument that can be controlled (e.g., with different configurations or settings) or may represent a set of fixed sensors from which to choose (e.g., a seismic, radar, and induction sensor for landmine detection, as discussed in [50]). Notice that detection (i.e., hypothesis testing) is a special case of our problem because it reduces the case where there are two classes: present and absent.

After each measurement is taken, we have to choose whether or not to produce a classification (i.e., an estimate $\hat{c} \in \mathcal{C}$). If we choose to produce such a classification, the scenario terminates. If not, we can continue to take another measurement by selecting a sensor control. The performance metric of interest here (to be maximized) is the probability of correct classification minus the total cost of sensors used.

To formulate this problem as a POMDP, we must specify the ingredients described in Section III-E: states, actions, state-transition law, reward function, observations, and observation law.

States. The possible states in our POMDP formulation of this classification problem are the possible classes, together with an extra state to represent that the scenario has terminated, which we will denote by τ. Therefore, the state space is given by $\mathcal{C} \cup \{\tau\}$. Note that the state changes only when we choose to produce a classification, as we will specify in the state-transition law below.

Actions. The actions here are of two kinds: we can either choose to take a measurement, in which case the action is the sensor control $u \in \mathcal{U}$, or we can choose to produce a classification, in which case the action is the class $\hat{c} \in \mathcal{C}$. Hence, the action space is given by $\mathcal{U} \cup \mathcal{C}$.

State-transition law. The state-transition law represents how the state evolves at each time step as a function of the action. As pointed out before, as long as we are taking measurements, the state does not change (because it represents the unknown object class). As soon as we choose to produce a classification, the state changes to the terminal state τ. Therefore, the state-transition law P_{trans} is given by

$$P_{\text{trans}}(x'|x,a) = \begin{cases} 1 & \text{if } a \in \mathcal{U} \text{ and } x' = x \\ 1 & \text{if } a \in \mathcal{C} \text{ and } x' = \tau \\ 0 & \text{otherwise.} \end{cases}$$

Reward function. The reward function R here is given by

$$R(x,a) = \begin{cases} -\kappa(a) & \text{if } a \in \mathcal{U} \text{ and } x \neq \tau \\ 1 & \text{if } a \in \mathcal{C} \text{ and } x = a \\ 0 & \text{otherwise.} \end{cases}$$

If we produce a classification, then the reward is 1 if the classification is correct, and otherwise it is 0. Hence, the mean of the reward when producing a classification is the probability that the classification is correct. If we use the finite-horizon objective function with horizon H, then the objective function represents the probability of producing a correct classification within the time horizon of H (e.g., representing some maximum time limit for producing a classification) minus the total sensing cost.

Observations. The observations in this problem represent the sensor outputs (measurements). The observation space is therefore the set of possible measurements.

Observation law. The observation law specifies the distribution of the observations given the state and action. So, if $x \in \mathcal{C}$ and $a \in \mathcal{U}$, then the observation law is given by $P_{\text{sensor}}(\cdot|x,a)$. If $x = \tau$, then we can define the observation law arbitrarily, because it does not affect the solution to the problem (recall that after the scenario terminates, represented by being in state τ, we no longer take any measurements).

Note that as long as we are still taking measurements and have not yet produced a classification, the belief state for this problem represents the *a posteriori* distribution of the unknown class being estimated. It is straightforward to show that the optimal policy for this problem will always produce a classification that maximizes the *a posteriori* probability (i.e., is a "MAP" classifier). However, it is not straightforward to deduce exactly when we should continue to take measurements and when we should produce a classification. Determining such an optimal policy requires solving the POMDP.

G. POMDP for an Adaptive Tracking Problem

We now consider a simple tracking problem and show how to formulate it using a POMDP framework. Our problem in illustrated in Figure 4. We have a Markov chain with state space $\mathcal{S}$ evolving according to a state-transition law given by T (i.e., for $s, s' \in \mathcal{S}$, $T(s'|s)$ is the probability of transitioning to state s' given that the state is s). We assume that $\mathcal{S}$ is a metric space—there is a function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ such that $d(s, s')$ represents a "distance" measure between s and s'.[1] The states of this Markov chain are not directly accessible—they represent quantities to be tracked over time (e.g., the coordinates and velocities of targets).

To do the tracking, as in the last section, we exploit measurements from a "controlled sensor" over time. At each time step, we first have to choose a control $u \in \mathcal{U}$. For each chosen control u, we get a measurement whose distribution depends on the Markov chain state s and control u, denoted $P_{\text{sensor}}(\cdot|s, u)$ as before (again, we assume that sensor measurements over time are independent). Each time we apply control u, we incur a cost of $\kappa(u)$ (i.e., as in the last example, the cost of using the controlled sensor depends on the control applied). As in the last example, the controlled sensor may represent a particular measurement instrument that can be controlled (e.g., with different configurations or settings) or may represent a set of fixed sensor assets from which to choose (e.g., multiple sensors distributed over a geographical region, where the control here is which subset of sensors to activate, as in [22], [23], [27], [40], [41]).

Each measurement is fed to a tracker, which is an algorithm that produces an estimate $\hat{s}_k \in \mathcal{S}$ of the state at each time k. For example, the tracker could be a Kalman filter or a particle filter [46]. The tracker has an internal state, which we will denote $z_k \in \mathcal{Z}$. The internal state is updated as a function of measurements:

$$z_{k+1} = f_{\text{tracker}}(z_k, y_k),$$

where y_k is the measurement generated at time k as a result of control u_k (i.e., if the Markov chain state at time k is s_k, then y_k has distribution $P_{\text{sensor}}(\cdot|s_k, u_k)$). The estimate $\hat{s}_k$ is a function of this internal state z_k. For example, in the case of a Kalman filter, the internal state represents a mean vector together with a covariance matrix. The output $\hat{s}_k$ is usually simply the mean vector. In the case of a particle filter, the internal state represents a set of particles. See [46] for explicit equations to represent f_{tracker}.

The performance metric of interest here (to be maximized) is the negative mean of the sum of the cumulative tracking error and the sensor usage cost over a horizon of H time steps. To be precise, the tracking error at time k is the "distance" between the output of the tracker, $\hat{s}_k$, and the true Markov chain state, s_k. Recall that the "distance" here is well-defined because we have assumed that $\mathcal{S}$ is a metric space. So the tracking error at time k is $d(\hat{s}_k, s_k)$.

As in the last section, to formulate this adaptive tracking problem as a POMDP, we must specify the ingredients described in Section III-E: states, actions, state-transition law, reward function, observations, and observation law.

States. It might be tempting to define the state space for this problem simply to be the state space for the Markov chain, $\mathcal{S}$. However, it is important to point out that the tracker also contains an internal state, and the POMDP state should take both into account. Accordingly, for this problem we will take the state

[1]For the case where $\mathcal{S}$ represents target kinematic states in Cartesian coordinates, we typically use the Euclidean norm for this metric.

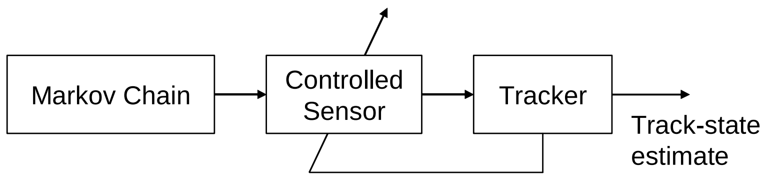


Fig. 4. An adaptive tracking system.

at time k to be the pair $[s_k, z_k]$, where s_k is the state of the Markov chain to be tracked, and z_k is the tracker state. Hence, the state space is $\mathcal{S} \times \mathcal{Z}$.

Actions. The actions here are the controls applied to the controlled sensor. Hence, the action space is simply $\mathcal{U}$.

State-transition law. The state-transition law specifies how the state changes at each time k, given the action a_k at that time. Recall that the state at time k is the pair $[s_k, z_k]$. The Markov chain state s_k makes a transition according to the transition probability $T(\cdot|s_k)$. The tracker state z_k makes a transition depending on the observation y_k. In other words, the transition distribution for the next tracker state given z_k is the distribution of $f_{\text{tracker}}(z_k, y_k)$ (which in turn depends on the measurement distribution $P_{\text{sensor}}(\cdot|s_k, a_k)$). This completely specifies the distribution of $[s_{k+1}, z_{k+1}]$ as a function of $[s_k, z_k]$ and a_k.

Reward function. The reward function is given by

$$R([s_k, z_k], a_k) = -(d(\hat{s}_k, s_k) + \kappa(a_k)),$$

where the reader should recall that the tracker output $\hat{s}_k$ is a function of z_k. Notice that the first term in the (per-time-step) reward, which represents tracking error, is not a function of a_k. Instead, the tracking errors depend on the actions applied over time through the track estimates $\hat{s}_k$ (which in turn depend on the actions through the distributions of the measurements).

Observations. As in the previous example, the observations here represent the sensor outputs (measurements). The observation space is therefore the set of possible measurements.

Observation law. The observation law is given by the measurement distribution $P_{\text{sensor}}(\cdot|s_k, a_k)$. Note that the observation law does not depend on z_k, the tracker state, even though z_k is part of the POMDP state.

H. POMDP for Motivating Example

In this section, we give mathematical forms for each of the ingredients listed in Section III-E for the motivating example described in Section II (these also apply to the simulation example in Section VII). To review, the motivating example dealt with an airborne sensor charged with detecting and tracking multiple moving targets. The airborne sensor is agile in that it can steer its beam to different ground locations. Each interrogation of the ground results in an observation as to the absence or presence of targets in the vicinity. The adaptive sensing problem is to use the collection of measurements made up to the current time to determine the best place to point next.

States. In this motivating problem, we are detecting and tracking N moving ground targets. For the purposes of this discussion we assume that N is known and fixed, and that the targets are moving in 2 dimensions (a more general treatment, where the number of targets is both unknown and time varying, is given elsewhere [35]). We denote these positions as $x_1, \ldots, x_N$ where x_i is a 2-dimensional vector corresponding to target i. Furthermore, because of the terrain, the position of the sensor influences the visibility of certain locations on the ground, so sensor position is an important component of the state. Denote the (directly observable) 3-dimensional sensor position by σ. Then the state space $\mathcal{X}$ consists of real-valued vectors in $\mathbb{R}^{2N+3}$, i.e., each state takes the form

$$x = [x_1, x_2, \ldots, x_{N-1}, x_N, \sigma].$$

Although not explicitly shown here, the surveillance region topology is assumed known and considered part of the problem specification. This specification affects the observation law, as we shall see below.

Actions. The airborne sensor is able to measure a single detection cell and make an imperfect measurement as to the presence or absence of a target in that cell. Therefore, the action $a \in \{1, \ldots, C\}$ is an integer specifying which of the C discrete cells is measured.

State-transition law. The state-transition law describes the distribution of the next state vector $x' = [x'_1, x'_2, \ldots, x'_N, \sigma']$ conditioned on the current state vector $x = [x_1, x_2, \ldots, x_N, \sigma]$ and the action a. Because our states are vectors in $\mathbb{R}^{2N+3}$, we will specify the state-transition law as a conditional density function. For simplicity, we have chosen to model the evolution of each of the N targets as independent and following a Gaussian law, i.e.,

$$T_{\text{single target}}(x'_i|x_i) = \frac{1}{2\pi|\Sigma|^{-1/2}} \exp^{-\frac{1}{2}(x_i - x'_i)^\top \Sigma^{-1}(x_i - x'_i)}, \quad i = 1, \ldots, N$$

(where x_i and x'_i are treated here as column vectors). In other words, each target moves according to a random walk (purely diffusive). Because of our independence assumption, we can write the joint target-motion law as

$$T_{\text{target}}(x'_1, \ldots, x'_N|x_1, \ldots, x_N) = \prod_{i=1}^{N} T_{\text{single target}}(x'_i|x_i).$$

The temporal evolution of the sensor position is assumed deterministic and known precisely (i.e., the aircraft if flying a pre-planned pattern). We use $f(\sigma)$ to denote the sensor trajectory function, which specifies the next position of the sensor given current sensor position σ; i.e., if the current sensor position is σ, then $f(\sigma)$ is exactly the next sensor position. Then, the motion law for the sensor is

$$T_{\text{sensor}}(\sigma'|\sigma) = \delta\big(\sigma' - f(\sigma)\big).$$

With these assumptions, the state-transition law is completely specified by

$$P_{\text{trans}}(x'|x, a) = T_{\text{target}}(x'_1, \ldots, x'_N|x_1, \ldots, x_N) T_{\text{sensor}}(\sigma'|\sigma).$$

Note that according to our assumptions, the actions taken do not affect the state evolution. In particular, we assume that the targets do not know they are under surveillance and consequently they do not take evasive action (see [31] for a model that includes evasion).

Reward function. In previous work [34], we have found that *information gain* provides a useful metric that captures a wide variety of goals. Information gain is a metric that measures the relative information increase between a prior belief state and a posterior belief state, i.e., it measures the benefit a particular observation has yielded. An information theoretic metric is intuitively pleasing as it measures different types of benefits (e.g., information about the number of targets present versus information about the positions of individual targets) on an equal footing, that of information gain. Furthermore, it has been shown that information gain can be viewed as a near universal proxy for any risk function [33]. Therefore, the reward used in this application is the gain in information between the belief state before a measurement b_k and the (measurement updated) belief state after a measurement is made $\hat{b}_k$. We use a particular information metric called the Rènyi divergence, defined as follows. The Rènyi divergence of two belief states p and q is given by

$$D_\alpha(p||q) = \frac{1}{\alpha - 1} \ln \sum_{x \in \mathcal{X}} p(x)^\alpha q(x)^{1-\alpha}$$

where $\alpha > 0$. To define the reward $r(b, a)$ in our context, given a belief state b and an action a, we first write,

$$\Delta_\alpha(b, a, y) = D_\alpha(\hat{b}||b),$$

where y is an observation with distribution given by the observation law $P_{\text{obs}}(\cdot|b, a)$ and $\hat{b}$ is the "updated" belief state computed as described earlier in Section III-B using Bayes' rule and knowledge of b, a, and y. Note that $\Delta_\alpha(b,a,y)$ is a random variable because it is a function of the random observation y, and hence its distribution depends on a. We will call this random variable the *myopic information gain*. The reward function is defined in terms of the myopic information gain by taking expectation: $r(b,a) = \mathrm{E}[\Delta_\alpha(b,a,y)|b,a]$.

Observations. When a cell is interrogated, the sensor receives return energy and thresholds this energy to determine whether it is to be declared a detection or a nondetection. This imperfect measurement gives evidence as to the presence or absence of targets in the cell. Additionally, the current sensor position is directly observable. Therefore, the observation is given by $[z,\sigma]$, where $z \in \{0,1\}$ is the one-bit observation representing detection or nondetection, and σ is the position of the sensor.

Observation law. Detection/nondetection is assumed to result from thresholding a Rayleigh-distributed random variable that characterizes the energy returned from an interrogation of the ground. The performance is completely specified by a probability of detection P_{d} and a false alarm rate P_{f}, which under the Rayleigh assumption are linked by a signal-to-noise-plus-clutter ratio, $SNCR$, by $P_{\mathrm{d}} = P_{\mathrm{f}}^{1/(1+SNCR)}$.

To precisely specify the observation model, we make the following notational definitions. First, let $o_a(x_1,\ldots,x_N)$ denote the occupation indicator function for cell a, defined as $o_a(x_1,\ldots,x_N) = 1$ when at least one of the targets projects into sensor cell a (i.e., at least one of the x_i locations are within cell a), and $o_a(x_1,\ldots,x_N) = 0$ otherwise. Furthermore, let $v_a(\sigma)$ denote the visibility indicator function for cell a, defined as $v_a(\sigma) = 1$ when cell a is visible from a sensor positioned at σ (i.e., there is no line of sight obstruction between the sensor and the cell), and $v_a(\sigma) = 0$ otherwise. Then the probability of receiving a detection given state $x = [x_1,\ldots,x_N,\sigma]$ and action a is

$$P_{\mathrm{det}}(x,a) = \begin{cases} P_{\mathrm{d}} & \text{if } o_a(x_1,\ldots,x_N)v_a(\sigma) = 1 \\ P_{\mathrm{f}} & \text{if } o_a(x_1,\ldots,x_N)v_a(\sigma) = 0. \end{cases}$$

Therefore, the observation law is specified completely by

$$P_{\mathrm{obs}}(z|x,a) = \begin{cases} P_{\mathrm{det}}(x,a) & \text{if } z = 1 \\ 1 - P_{\mathrm{det}}(x,a) & \text{if } z = 0. \end{cases}$$

IV. Basic Principle: Q-Value Approximation

A. Overview and History

In this section, we describe the basic principle underlying approximate methods to solve adaptive sensing problems that are posed as POMDPs. This basic principle is due to Bellman, and gives rise to a natural framework in which to discuss a variety of approximation approaches. Specifically, these approximation methods all boil down to the problem of approximating Q-values.

Methods for solving POMDPs have their roots in the field of optimal control, which dates back to the end of the seventeenth century with the work of Johann Bernoulli [58]. This field received significant interest in the middle of the twentieth century, when much of the modern methodology was developed, most notably by Bellman [3], who applied *dynamic programming* to bear on optimal control, and Pontryagin [44], who introduced his celebrated *maximum principle* based on calculus of variations. Since then, the field of optimal control has enjoyed much fruit in its application to control problems arising in engineering and economics.

The recent history of methods to solve optimal stochastic decision problems took an interesting turn in the second half of the twentieth century with the work of computer scientists in the field of artificial intelligence seeking to solve “planning” problems (roughly analogous to what engineers and economists call optimal control problems). The results of their work most relevant to the POMDP methods discussed here are reported in a number of treatises from the 80s and 90s [14], [29], [61], [30]. The methods developed in the artificial intelligence (machine learning) community aim to provide computationally feasible approximations to optimal solutions for complex planning problems under uncertainty. The operations research literature has also continued to reflect ongoing interest in computationally feasible methods for optimal decision problems [39], [11], [45].

The connection between the significant work done in the artificial intelligence community and those of the earlier work on optimal control is noted by Bertsekas and Tsitsiklis in their 1996 book [4]. In particular, they note that the developments in *reinforcement learning*—the approach taken by artificial intelligence researchers for solving planning problems—is most appropriately understood in the framework of Markov decision theory and dynamic programming. This framework is now widely reflected in the artificial intelligence literature [29], [61], [30], [54]. Our treatment in this paper rests on this firm and rich foundation (though our focus is not on reinforcement learning methods).

B. Bellman's Principle and Q-values

The key result in Markov decision theory relevant here is Bellman's principle. Let $V_H^*(b_0)$ be the optimal objective function value (over horizon H) with b_0 as the initial belief state. Then, *Bellman's principle* states that

$$V_H^*(b_0) = \max_a (r(b_0, a) + \mathrm{E}[V_{H-1}^*(b_1)|b_0, a])$$

where b_1 is the random next belief state (with distribution depending on a), and $\mathrm{E}[\cdot|b_0, a]$ represents conditional expectation with respect to the random next state b_1, whose distribution depends on b_0 and a. Moreover,

$$\pi_0^*(b_0) = \arg\max_a (r(b_0, a) + \mathrm{E}[V_{H-1}^*(b_1)|b_0, a])$$

is an optimal policy.

Define the *Q-value* of taking action a at state b_k as

$$Q_{H-k}(b_k, a) = r(b_k, a) + \mathrm{E}[V_{H-k-1}^*(b_{k+1})|b_k, a],$$

where b_{k+1} is the random next belief state (which depends on the observation y_k at time k, as described in Section III-B). Then, Bellman's principle can be rewritten as

$$\pi_k^*(b_k) = \arg\max_a Q_{H-k}(b_k, a)$$

i.e., the optimal action at belief-state b_k (at time k, with a horizon-to-go of $H - k$) is the one with largest Q-value at that belief state. This principle, called *lookahead*, is the heart of POMDP solution approaches.

C. Stationary Policies

In general, an optimal policy is a function of time k. If H is sufficiently large, then the optimal policy is approximately *stationary* (independent of k). This is intuitively clear: if the end of the time horizon is a million years away, then how we should act today given a belief-state is the same as how we should act tomorrow with the same belief state. Said differently, if H is sufficiently large, the difference between Q_H and Q_{H-1} is negligible. Moreover, if needed we can always incorporate time itself into the definition of the state, so that dependence on time is captured simply as dependence on state.

Henceforth we will assume for convenience there is a stationary optimal policy, and this is what we seek. We will use the notation π for stationary policies (with no subscript k)—this significantly simplifies the notation. Our approach is equally applicable to the short-horizon, nonstationary case, with appropriate notational modification (to account for the time dependence of decisions).

D. Receding Horizon

Assuming H is sufficiently large and that we seek a stationary optimal policy, at any time k we write:

$$\pi^*(b) = \arg\max_a Q_H(b, a).$$

Notice that the horizon is taken to be fixed at H, regardless of the current time k. This is justified by our assumption that H is so large that at any time k, the horizon is still approximately H time steps away. This approach of taking the horizon to be fixed at H is called *receding horizon control*. For convenience, we will also henceforth drop the subscript H from our notation (unless the subscript is explicitly needed).

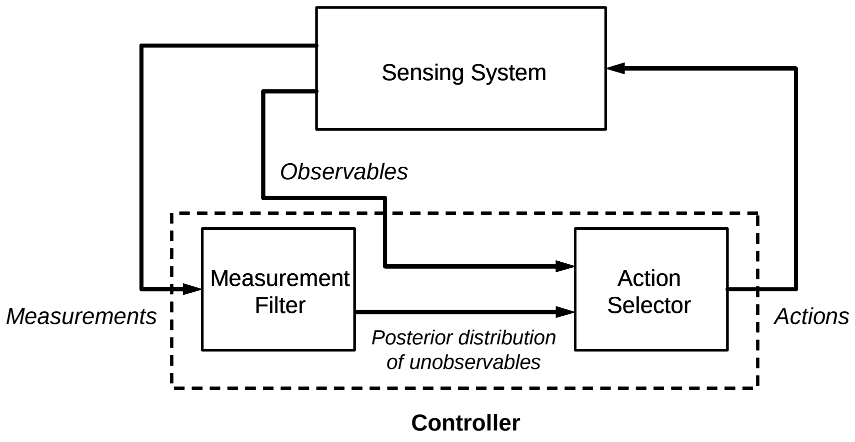


Fig. 5. Basic lookahead framework.

E. Approximating Q-values

Recall $Q(b, a)$ is the reward $r(b, a)$ of taken action a at belief-state b plus the expected cumulative reward of applying the optimal policy for all future actions. This second term in the Q-value is in general difficult to obtain, especially when the belief-state is large. For this reason, approximation methods are necessary to obtain Q-values. Note that the quality of an approximation is not so much in the accuracy of the actual Q-values obtained, but in the *ranking* of the actions reflected by their *relative* values.

In Section VI, we describe a variety of methods to approximate Q-values. But before discussing such methods, we first describe the basic control framework for using Q-values to inform control decisions.

V. Basic Control Architecture

By Bellman's principle, knowing the Q-values allows us to make optimal control decisions. In particular, if we are currently at belief-state b, we need only find the action a with the largest $Q(b, a)$. This principle yields a basic control framework that is illustrated in Figure 5. The top-most block represents the sensing system, which we treat as having an input and two forms of output. The input represents actions (external control commands) we can apply to control the sensing system. Actions usually include sensor-resource controls, such as which sensor(s) to activate, at what power level, where to point, what waveforms to use, and what sensing modes to activate. Actions may also include communication-resource controls, such as the data rate for transmission from each sensor.

The two forms of outputs from the sensing system represent:

1) Fully observable aspects of the internal state of the sensing system (called *observables*), and
2) Measurements (observations) of those aspects of the internal state that are not directly observable (which we refer to simply as *measurements*).

We assume that the underlying state-space is the Cartesian product of two sets, one representing unobservables and the other representing observables. Target states are prime examples of unobservables. So, measurements are typically the outputs of sensors, representing observations of target states. Observables include things like sensor locations and orientations, which sensors are activated, battery status readings, etc. In the remainder of this section, we describe the components of our control framework. Our description starts from the architecture of Figure 5 and progressively fills in the details.

A. Controller

At each decision epoch, the *controller* takes the outputs (measurements and observables) from the sensing system and, in return, generates an action that is fed back to the sensing system. This basic closed-loop architecture is familiar to mainstream control system design approaches.

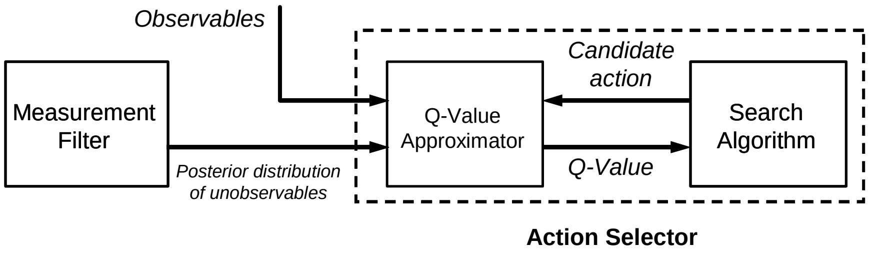


Fig. 6. Basic components of the action selector.

The controller has two main components. The first is the *measurement filter*, which takes as input the measurements, and provides as output the *a posteriori* (posterior) distribution of unobservable internal states (henceforth called *unobservables*). In the typical situation where the unobservables are target states, the measurement filter outputs a posterior distribution on target states given the measurement history. The measurement filter is discussed further below. The posterior distribution of the unobservables in addition to the observables form the belief state, the posterior distribution of the underlying state. The second component is the *action selector*, which takes the belief state and computes an action (the output of the controller). The basis for action selection is Bellman's principle, using Q-values. This is discussed below.

B. Measurement Filter

The measurement filter computes the posterior distribution given measurements. This component is present in virtually all target-tracking systems. It turns out that the posterior distribution can be computed iteratively: each time we obtain a new measurement, the posterior distribution can be obtained by updating the previous posterior distribution based on knowing the current action, the transition law, and the observation law. This update is based on Bayes' rule, described earlier in Section III-B.

The measurement filter can be constructed in a number of ways. If the posterior distribution always resides within a family of distributions that is conveniently parameterized, then all we need to do is keep track of the belief-state parameters. This is the case, for example, if the belief state is Gaussian. Indeed, if the unobservables evolve in a linear fashion, then these Gaussian parameters can be updated using a Kalman filter. In general, however, it is not practical to keep track of the exact belief state. Indeed, a variety of options have been explored for belief-state representation and simplification (e.g., [49], [48], [60]). We will have more to say about belief-state simplification in Section VI-K.

Particle filtering is a Monte Carlo sampling method for updating posterior distributions. Instead of maintaining the exact posterior distribution, we maintain a set of representative samples from that distribution. It turns out that this method dovetails naturally with Monte Carlo sampling-based methods for Q-value approximation, as we will describe later in Section VI-H.

C. Action Selector

As shown in Figure 6, the action selector consists of a search (optimization) algorithm that optimizes an objective function, the Q-*function*, with respect to an action. In other words, the Q-function is a function of the action—it maps each action, at a given belief state, to its Q-value. The action that we seek is one that maximizes the Q-function. So, we can think of the Q-function as a kind of "action-utility" function that we wish to maximize. The search algorithm iteratively generates a candidate action and evaluates the Q-function at this action (this numerical quantity is the Q-value), searching over the space of candidate actions for one with the largest Q-value. Methods for obtaining (approximating) the Q-values is described in the next section.

VI. Q-Value Approximation Methods

A. Basic Approach

Recall the definition of the Q-value,

$$Q(b,a) = r(b,a) + \mathrm{E}[V^*(b')|b,a], \tag{1}$$

where b' is the random next belief state (with distribution depending on a). In all but very special problems, it is impossible to compute the Q-value exactly. In this section, we describe a variety of methods to approximate the Q-value. Because the first term on the right-hand side of (1) is usually easy to compute, most approximation methods focus on the second term. As pointed out before, it is important to realize that the quality of an approximation to the Q-value is not so much in the accuracy of the actual values obtained, but in the *ranking* of the actions reflected by their *relative* values.

We should point out that each of the approximation methods presented in this section has its own domain of applicability. Traditional reinforcement learning approaches (Section VI-F), predicated on running a large number of simulations to "train," are broadly applicable as they only require a generative model. However, these methods often have infeasible computational burden owing to the long training time required for some problems. Furthermore, there is an extensibility problem, where a trained function may perform very poorly if the problem changes slightly between the training stage and the application stage. To address these concerns, we present several sampling techniques (Sections VI-B, VI-H, VI-I, VI-K) which are also very broadly applicable as they only require a generative model. These methods do not require a training phase, per se, but do on-line estimation. However, in some instances, these too may require more computations than desirable. Similarly, parametric approximations (Section VI-E) and action-sequence approximations (Section VI-G) are general in applicability but may entail excessive computational requirements. Relaxation methods (Section VI-C) and heuristics (Section VI-D) may provide reduced computation but require advanced domain knowledge.

B. Monte Carlo Sampling

In general, we can think of Monte Carlo methods simply as the use of computer generated random numbers in computing expectations of random variables through averaging over many samples. With this in mind, it seems natural to consider using Monte Carlo methods to compute the value function directly based on Bellman's equation:

$$V_H^*(b_0) = \max_{a_0}(r(b_0,a_0) + \mathrm{E}[V_{H-1}^*(b_1)|b_0,a_0]).$$

Notice that the second term on the right-hand side involves expectations (one per action candidate a_0), which can be computed using Monte Carlo sampling. However, the random variable inside each expectation is itself an objective function value (with horizon $H-1$), and so it too involves a max of an expectation via Bellman's equation:

$$V_H^*(b_0) = \max_{a_0}\left(r(b_0,a_0) + \mathrm{E}\left[\max_{a_1}(r(b_1,a_1) + \mathrm{E}[V_{H-2}^*(b_2)|b_1,a_1])\middle| b_0,a_0\right]\right).$$

Notice we now have two "layers" of max and expectation, one "nested" within the other. Again, we see the inside expectation involves the value function (with horizon $H-2$), which again can be written as a max of expectations. Proceeding this way, we can write $V_H^*(b_0)$ in terms of H layers of max and expectations. Each expectation can be computed using Monte Carlo sampling. The remaining question is how computationally burdensome is this task?

Kearns, Mansour, and Ng [28] have provided a method to calculate the computational burden of approximating the value function using Monte Carlo sampling as described above, given some prescribed accuracy in the approximation of the value function. Unfortunately, it turns out that for practical POMDP problems this computational burden is prohibitive, even for modest degrees of accuracy. So, while

Bellman’s equation suggests a natural Monte Carlo method for approximating the value function, the method is not useful in practice. For this reason, we seek alternative approximation methods. In the next few subsections, we explore some of these methods.

C. Relaxation of Optimization Problem

Some problems that are difficult to solve become drastically easier if we *relax* certain aspects of the problem. For example, by removing a constraint in the problem, the “relaxed” problem may yield to well-known solution methods. This constraint relaxation enlarges the constraint set, and so the solution obtained may no longer be feasible in the original problem. However, the objective function value of the solution *bounds* the optimal objective function value of the original problem.

The Q-value involves the quantity $V^*(b')$, which can be viewed as the optimal objective function value corresponding to some optimization problem. The method of relaxation, if applicable, gives rise to a bound on $V^*(b')$, which then provides an approximation to the Q-value. For example, a relaxation of the original POMDP may result in a bandit problem (see [36], [37]), or may be solvable via linear programming (see [18], [19]). (See also specific applications to sensor management [10], [56].) In general, the quality of this approximation is a function of the specific relaxation and is very problem specific. For example, Castanon [10] suggests that in his setting his relaxation approach is feasible for generating near-optimal solutions. Additionally, Washburn et al. [56] show that the performance of their index rule is eclipsed by that of multi-step lookahead under certain conditions of the process noise, while being much closer in the low-noise situation. While it is sometimes possible to apply analytical approaches to a relaxed version of the problem, it is generally accepted that problems that can be posed as POMDPs are unlikely to be amenable to analytical solution approaches.

Bounds on the optimal objective function value can also be obtained by approximating the state space. Lovejoy [38] shows how to approximate the state space by a finite grid of points, and use that grid to construct upper and lower bounds on the optimal objective function.

D. Heuristic Approximation

In some applications we are unable to compute Q-values directly, but can use domain knowledge to develop an idea of its behavior. If so, we can heuristically construct a Q-function based on this knowledge.

Recall from (1) that the Q-value is the sum of two terms, where the first term (the immediate reward) is usually easy to compute. Therefore, it often suffices to approximate only the second term in (1), which is the mean optimal objective function value starting at the next belief state, which we call the *expected value-to-go* (EVTG). (Note the EVTG is a function of both b and a, because the distribution of the next belief state is a function of b and a.) In some problems, it is possible to construct a heuristic EVTG based on domain knowledge. If the constructed EVTG properly reflects tradeoffs in the selection of alternative actions, then the ranking of these actions via their Q-values will result in the desired “lookahead.”

For example, consider the motivating example of tracking multiple targets with a single sensor. Suppose we can only measure the location of one target per decision epoch. The problem then is to decide which location to measure and the objective function is the aggregate (multi-target) tracking error. The terrain over which the targets are moving is such that the measurement errors are highly location dependent, for example because of the presence of topological features which cause some areas to be invisible from a future sensor position. In this setting, it is intuitively clear that if we can predict sensor and target motion so that we expect a target is about to be obscured, then we should focus our measurements on that target immediately before the obscuration so that its track accuracy is improved and the overall tracking performance maximized in light of the impending obscuration.

The same reasoning applies in a variety of other situations, including those where targets are predicted to become unresolvable to the sensor (e.g., two targets that cross) or where the target and sensor motion is such that future measurements are predicted to be less reliable (e.g., a bearings-only sensor that is moving away from a target). In these situations, we advocate a heuristic method that replaces the EVTG by a function that captures the long-term benefit of an action in terms of an “opportunity cost” or “regret.” That is, we approximate the Q-value as

$$Q(b,a) \approx r(b,a) + wN(b,a)$$

where $N(b,a)$ is an easily computed heuristic approximation of the long-term value, and w is a weighting term that allows us to trade the influence of the immediate value and the long-term value. As a concrete example of a useful heuristic, we have used the “gain in information for waiting” as a choice of $N(b,a)$ [32]. Specifically, let $\bar{g}_a^k$ denote the expected value of the Rényi divergence between the belief state at time k and the updated belief state at time k after taking action a, as defined in Section III-H (i.e., the *myopic information gain*). Note that this myopic information gain is a random variable whose distribution depends on a, as explained in Section III-H. Let $p_a^k(\cdot)$ denote the distribution of this random variable. Then a useful approximation of the long-term value of taking action a is the gain (loss) in information received by waiting until a future time step to take the action,

$$N(b,a) \approx \sum_{m=1}^{M} \gamma^m \operatorname{sgn}\bigl(\bar{g}_a^k - \bar{g}_a^{k+m}\bigr) D_\alpha\bigl(p_a^k(\cdot)||p_a^{k+m}(\cdot)\bigr)$$

where M is the number of time steps in the future that are considered.

Each term in the summand of $N(b,a)$ has two components. First, $\operatorname{sgn}\bigl(\bar{g}_a^k - \bar{g}_a^{k+m}\bigr)$ signifies if the expected reward for taking action a in the future is more or less than the present. A negative value implies that the future is better and that the action ought to be discouraged at present. A positive value implies that the future is worse and that the action ought to be encouraged at present. This may happen, for example, when the visibility of a given target is getting worse with time. The second term, $D_\alpha\bigl(p_a^k(\cdot)||p_a^{k+m}(\cdot)\bigr)$, reflects the magnitude of the change in reward using the divergence between the density on myopic rewards at the current time step and at a future time step. A small number implies the present and future rewards are very similar, and therefore the nonmyopic term should have little impact on the decision making.

Therefore, $N(b,a)$ is positive if an action is less favorable in the future (e.g., the target is about to become obscured). This encourages taking actions that are beneficial in the long term, and not just taking actions based on their immediate reward. Likewise, the term is negative if the action is more favorable in the future (e.g., the target is about to emerge from an obscuration). This discourages taking actions now that will have more value in the future.

E. Parametric Approximation

In situations where a heuristic Q-function is difficult to construct, we may consider methods where the Q-function is approximated by a parametric function (by this we mean that we have a function approximator parameterized by one or more parameters). Let us denote this approximation by $\tilde{Q}(b,\theta)$, where θ is a parameter (to be tuned appropriately). For this approach to be useful, the computation of $\tilde{Q}(b,\theta)$ has to be relatively simple, given b and θ. Typically, we seek approximations for which it is easy to set the value of the parameter θ appropriately, given some information of how the Q-values “should” behave (e.g., from expert knowledge, empirical results, simulation, or on-line observation). This adjustment or tuning of the parameter θ is called *training*. In contrast to on-line approximation methods discussed in this section, the training process in parametric approximation is often done off-line.

As in the heuristic approximation approach, the approximation of the Q-function by the parametric function approximator is usually accomplished by approximating the EVTG, or even directly approximating the objective function V^*.[2] In the usual parametric approximation approach, the belief state b is first mapped to a set of *features*. The features are then passed through a parametric function to approximate $V^*(b)$. For example, in the problem of tracking multiple targets with a single sensor, we may extract from the belief state some information on the location of each target relative to the sensor, taking into

[2]In fact, given a POMDP, the Q-value can be viewed as the objective function value for a related problem; see [4].

account the topology. These constitute features. For each target, we then assign a numerical value to these features, reflecting the measurement accuracy. Finally, we take a linear combination of these numerical values, where the coefficients of this linear combination serve the role of the parameters to be tuned.

The parametric approximation method has some advantages over methods based only on heuristic construction. First, the training process usually involves numerical optimization algorithms, and thus well-established methodology can be brought to bear on the problem. Second, even if we lack immediate expert knowledge on our problem, we may be able to experiment with the system (e.g., by using a simulation model). Such empirical output is useful for training the function approximator. Common training methods found in the literature go by the names of reinforcement learning, Q-learning, neurodynamic programming, and approximate dynamic programming. We have more to say about reinforcement learning in the next section.

The parametric approximation approach may be viewed as a systematic method to implement the heuristic approach. But note that even in the parametric approach, some heuristics are still needed in the choice of features and in the form of the function approximator. For further reading, see [4].

F. Reinforcement Learning

A popular method for approximating the Q-function based on the parametric approximation approach is *reinforcement learning* or *Q-learning* [57]. Recall that the Q-function satisfies the equation

$$Q(b,a) = r(b,a) + \mathrm{E}\left[\max_{\alpha} Q(b',\alpha) \middle| b,a \right]. \tag{2}$$

In Q-learning, the Q-function is estimated from multiple trajectories of the process. Assuming as usual that the number of states and actions are finite, we can represent $Q(b,a)$ as a lookup table. In this case, given an arbitrary initial value of $Q(b,a)$, the one-step Q-learning algorithm [53] is given by the repeated application of the update equation:

$$Q(b,a) \leftarrow (1-\beta)Q(b,a) + \beta\left(r(b,a) + \max_{\alpha} Q(b',\alpha) \right), \tag{3}$$

where β is a parameter in $(0,1)$ representing a “learning rate,” and each of the 4-tuples $\{b,a,b',r\}$ are examples of states, actions, next states, and rewards incurred during the training phase. With enough examples of belief states and actions, the Q-function can be “learned” via simulation or on-line.

Unfortunately, in most realistic problems (the problems considered in this paper included) it is infeasible to represent the Q-function as a lookup table. This is either due to the large number of possible belief states (our case), actions, or both. Therefore, as pointed out in the last section, function approximation is required. A standard and simplest class of Q-function approximators are linear combinations of basis functions (also called features):

$$Q(b,a) = \theta(a)^\top \phi(b), \tag{4}$$

where $\phi(b)$ is a feature vector (often constructed by a domain expert) associated with state b and the coefficients of $\theta(a)$ are to be estimated, i.e., the training data is used to learn the best approximation to $Q(b,a)$ among all linear combinations of the features. Gradient descent is used with the training data to update the estimate of $\theta(a)$:

$$\begin{aligned}\theta(a) &\leftarrow \theta(a) + \beta\left(r(b,a) + \max_{a'} Q(b',a') - Q(b,a) \right) \nabla_\theta Q(b,a) \\ &= \theta(a) + \beta\left(r(b,a) + \max_{a'} \theta(a')^\top \phi(b') - \theta(a)^\top \phi(b) \right) \phi(b).\end{aligned}$$

Note that we have taken advantage of the fact that for the case of a linear function approximator, the gradient is given by $\nabla Q(b,a) = \phi(b)$. Hence, at every iteration, $\theta(a)$ is updated in the direction that minimizes the empirical error in (2). When a lookup table is used in (4), this algorithm reduces to (3). Once the learning of the vector $\theta(a)$ is completed, optimal actions can be computed according to

$\arg\max_a \theta(a)^\top \phi(b)$. Determining the learning rate (β) and the number of training episodes required is a matter of active research.

Selecting a set of features that simultaneously provide both an adequate description of the belief state and a parsimonious representation of the state space requires domain knowledge. For the illustrative example that we use in this paper (see Section III-H), the feature vector $\phi(b)$ should completely characterize the surveillance region and capture its nonstationary nature. For consistency in comparison to other approaches, we appeal to features that are based on information theory, although this is simply one possible design choice. In particular, we use the expected myopic information gain at the current time step and the expected myopic information gain at the next time step as features which characterize the state. Specifically, let $r(b,a) = \mathrm{E}[\Delta_\alpha(b,a,y)|b,a]$ be defined as in Section III-H. Next, define b' to be the belief state at the hypothetical "next" time step starting at the current belief state b, computed using the second of the two-step update procedure in Section III-B. In other words, b' is what results in the next step if only a state transition takes place, without an update based on incorporating a measurement. Then, the feature vector is

$$\phi(b) = [r(b,1),\ldots,r(b,C),r(b',1),\ldots,r(b',C)]$$

where C is the number of cells (and also the number of actions). In the situation of time-varying visibility, these features capture the immediate value of various actions and allow the system to learn the long-term value by looking at the change in immediate value of the actions over time. In a more general version of this problem, actions might include more than just which cell to measure—for example, actions might also involve which waveform to transmit. In these more general cases, the feature vector will be have more components to account for the larger set of possible actions.

G. Action-Sequence Approximations

Let us write the value function (optimal objective function value as a function of belief state) as

$$\begin{aligned} V^*(b) &= \max_\pi \mathrm{E}\left[\left.\sum_{k=0}^{H-1} r(b_k,\pi(b_k))\right| b,\pi(b)\right] \\ &= \mathrm{E}\left[\left.\max_{a_0,\ldots,a_{H-1}:a_k=\pi(b_k)} \sum_{k=0}^{H-1} r(b_k,a_k)\right| b\right], \end{aligned} \tag{5}$$

where the notation $\max_{a_0,\ldots,a_{H-1}:a_k=\pi(b_k)}$ means maximization subject to the constraint that each action a_k is a (fixed) function of the belief state b_k. If we relax this constraint on the actions and allow them to be arbitrary random variables, then we have an upper bound on the value function:

$$\hat{V}_{\mathrm{HO}}(b) = \mathrm{E}\left[\left.\max_{a_0,\ldots,a_{H-1}} \sum_{k=0}^{H-1} r(b_k,a_k)\right| b\right].$$

In some applications, this upper bound provides a suitable approximation to the value function. The advantage of this method is that in certain situations the computation of the "max" above involves solving a relatively easy optimization problem. This method is called *hindsight optimization* [17], [59].

One implementation involves averaging over many Monte Carlo simulation runs to compute the expectation above. In this case, the "max" is computed for each simulation run by first generating all the random numbers for that run, and then applying a static optimization algorithm to compute optimal actions $a_0,\ldots,a_{H-1}$. It is easy now to see why we call the method "hindsight" optimization: the optimization of the action sequence is done after knowing all uncertainties over time, as if making decisions in hindsight.

As an alternative to relaxing the constraint in (5) (that each action a_k is a fixed function of the belief state b_k), suppose we further *restrict* each action to be simply fixed (not random). This restriction gives rise to a lower bound on the value function:

$$\hat{V}_{\mathrm{FO}}(b) = \max_{a_0,\ldots,a_{H-1}} \mathrm{E}[r(b_0,a_0)+\cdots+r(b_{H-1},a_{H-1})|b,a_0,\ldots,a_{H-1}].$$

To use analogous terminology to “hindsight optimization,” we call this method *foresight optimization*—we make decisions before seeing what actually happens, based on our expectation of what will happen. The method is also called *open loop feedback control* [6]. For a tracking application of this, see [15].

We should also point out some alternatives to the simple hindsight or foresight approaches above. In [60], more sophisticated bounds are described that do not involve simulation, but instead rely on convexity. The method in [43] also does not involve simulation, but approximates the future belief-state evolution using a single sample path.

H. Rollout

In this section, we describe the method of *policy rollout* (or simply *rollout*) [7]. The basic idea is simple. First let $V^\pi(b_0)$ be the objective function value corresponding to policy π. Recall that $V^* = \max_\pi V^\pi$. In the method of rollout, we assume that we have a candidate policy π_{base} (called the *base policy*), and we simply replace V^* in (1) by $V^{\pi_{\text{base}}}$. In other words, we use the following approximation to the Q-value:

$$Q^{\pi_{\text{base}}}(b,a) = r(b,a) + \mathrm{E}[V^{\pi_{\text{base}}}(b')|b,a].$$

We can think of $V^{\pi_{\text{base}}}$ as the performance of applying π_{base} in our system. In many situations of interest, $V^{\pi_{\text{base}}}$ is relatively easy to compute, either analytically, numerically, or via Monte Carlo simulation.

It turns out that the policy π defined by

$$\pi(b) = \arg\max_a Q^{\pi_{\text{base}}}(b,a) \tag{6}$$

is at least as good as π_{base} (in terms of the objective function); in other words, this step of using one policy to define another policy has the property of *policy improvement*. This result is the basis for a method known as *policy iteration*, where we iteratively apply the above policy-improvement step to generate a sequence of policies converging to the optimal policy. However, policy iteration is difficult to apply in problems with large belief-state spaces, because the approach entails explicitly representing a policy and iterating on it (remember that a policy is a mapping with the belief-state space $\mathcal{B}$ as its domain).

In the method of policy rollout, we do not explicitly construct the policy π in (6). Instead, at each time step, we use (6) to compute the output of the policy at the current belief-state. For example, the term $\mathrm{E}[V^{\pi_{\text{base}}}(b')|b,a]$ can be computed using Monte Carlo sampling. To see how this is done, observe that $V^{\pi_{\text{base}}}(b')$ is simply the mean cumulative reward of applying policy π_{base}, a quantity that can be obtained by Monte Carlo simulation. The term $\mathrm{E}[V^{\pi_{\text{base}}}(b')|b,a]$ is the mean with respect to the random next belief-state b' (with distribution that depends on b and a), again obtainable via Monte Carlo simulation. We provide more details in Section VI-J. In our subsequent discussion of rollout, we will focus on its implementation using Monte Carlo simulation. For an application of the rollout method to sensor scheduling for target tracking, see [22], [23], [27], [40], [41].

I. Parallel Rollout

An immediate extension to the method of rollout is to use multiple base policies. So suppose that $\Pi_B = \{\pi^1, \dots, \pi^n\}$ is a set of base policies. Then replace V^* in (1) by

$$\hat{V}(b) = \max_{\pi \in \Pi_B} V^\pi(b).$$

We call this method *parallel rollout* [12]. Notice that the larger the set Π_B, the tighter $\hat{V}(b)$ becomes as a bound on $V^*(b)$. Of course, if Π_B contains the optimal policy, then $\hat{V} = V^*$. It follows from our discussion of rollout that the policy improvement property also holds here. As with the rollout method, parallel rollout can be implemented using Monte Carlo sampling.

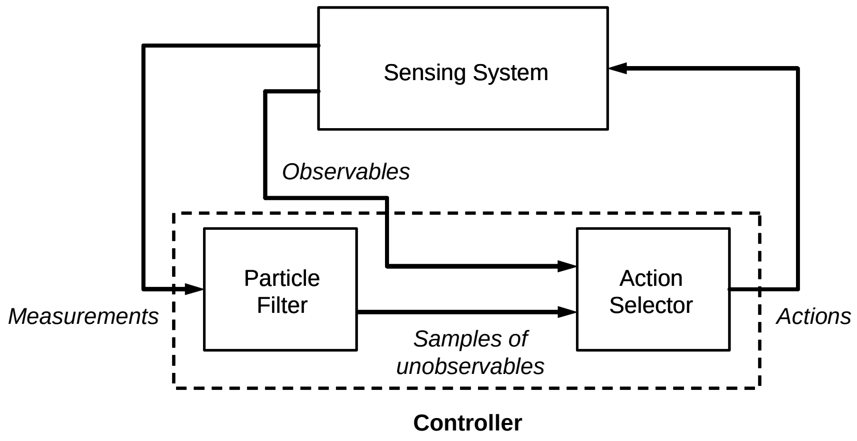


Fig. 7. Basic control architecture with particle filtering.

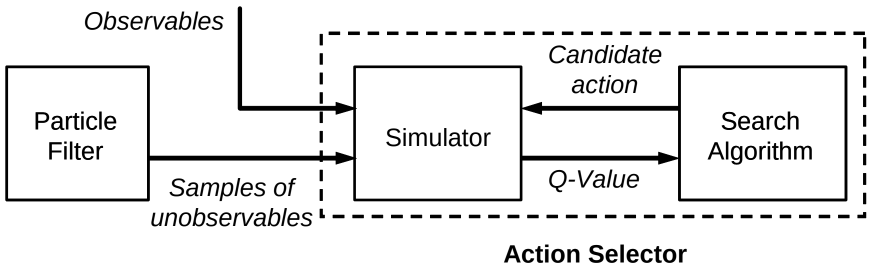


Fig. 8. Components of the action selector.

J. Control Architecture in the Monte Carlo Case

The method of rollout provides a convenient turnkey (systematic) procedure for Monte-Carlo-based decision making and control. Here, we specialize the general control architecture of Section V to the use of particle filtering for belief-state updating and a Monte Carlo method for Q-value approximation (e.g., rollout). We note that there is increasing interest in Monte Carlo methods for solving Markov decision processes [54], [11]. Particle filtering, which is a Monte Carlo sampling method for updating posterior distributions, dovetails naturally with Monte Carlo methods for Q-value approximation. An advantage of the Monte Carlo approach is that it does not rely on analytical tractability—it is straightforward in this approach to incorporate sophisticated models for sensor characteristics and target dynamics.

Figure 7 shows the control architecture specialized to the Monte Carlo setting. In contrast to Figure 5, a particle filter plays the role of the measurement filter, and its output consists of samples of the unobservables. Figure 8 shows the action selector in this setting. Contrasting this with Figure 6, we see that a Monte Carlo simulator plays the role of the Q-value approximator (e.g., via rollout). Search algorithms that are suitable here include the method of [51], which is designed for such problems, dovetails well with a simulation-based approach, and accommodates heuristics to guide the search within a rigorous framework.

As a specific example, consider applying the method of rollout. In this case, the evaluation of the Q-value for any given candidate action relies on a simulation model of the sensing system with some base policy. This simulation model is a “dynamic” model in that it evaluates the behavior of the sensing system over some horizon of time (specified beforehand). The simulator requires as inputs the current observables and samples of unobservables from the particle filter (to specify initial conditions) and a candidate action. The output of the simulator is a Q-value corresponding to the current measurements and observables, for the given candidate action. The output of the simulator represents the mean performance of applying the base policy, depending on the nature of the objective function. For example, the performance measure of the system may be the negative mean of the sum of the cumulative tracking error and the sensor usage cost over a horizon of H time steps, given the current system state and candidate action.

To elaborate on exactly how the Q-value approximation using rollout is implemented, suppose we are given the current observables and a set of samples of the unobservables (from the particle filter). The current observables together with a single sample of unobservables represent a candidate current underlying state of the sensing system. Starting from this candidate current state, we simulate the application of the given candidate action (which then leads to a random next state), followed by application of the base policy for the remainder of the time horizon—during this time horizon, the system state evolves according to the dynamics of the sensing system as encoded within the simulation model. For this single simulation run, we compute the "action utility" of the system (e.g., the negative of the sum of the cumulative tracking error and sensor usage cost over that simulation run). We do this for each sample of the unobservables, and then average over the performance values from these multiple simulation runs. This average is what we output as the Q-value.

The samples of the unobservables from the particle filter that are fed to the simulator (as candidate initial conditions for unobservables) may include all the particles in the particle filter (so that there is one simulation run per particle), or may constitute only a subset of the particles. In principle, we may even run multiple simulation runs per particle.

The above Monte Carlo method for approximating POMDP solutions has some beneficial features. First, it is flexible in that a variety of adaptive sensing scenarios can be tackled using the same framework. This is important because of the wide variety of sensors encountered in practice. Second, the method does not require analytical tractability; in principle, it is sufficient to simulate a system component, whether or not its characteristics are amenable to analysis. Third, the framework is modular in the sense that models of individual system components (e.g., sensor types, target motion) may be treated as "plug-in" modules. Fourth, the approach integrates naturally with existing simulators (e.g., Umbra [20]). Finally, the approach is inherently nonmyopic, allowing the tradeoff of short-term gains for long-term rewards.

K. Belief-State Simplification

If we apply the method of rollout to a POMDP, we need a base policy that maps belief states to actions. Moreover, we need to simulate the performance of this policy—in particular, we have to sample future belief states as the system evolves in response to actions resulting from this policy. Because belief states are probability distributions, keeping track of them in a simulation is burdensome.

A variety of methods are available to approximate the belief state. For example, we could simulate a particle filter to approximate the evolution of the belief state (as described previously), but even this may be unduly burdensome. As a further simplification, we could use a Gaussian approximation and keep track only of the mean and covariance of the belief state using a Kalman filter or any of its extensions, including *extended Kalman filters* and *unscented Kalman filters* [26]. Naturally, we would expect that the more accurate the approximation of the belief state, the more burdensome the computation.

An extreme special case of the above tradeoff is to use a Dirac delta distribution for belief states in our simulation of the future. In other words, in our lookahead simulation, we do away with keeping track of belief states altogether and instead simulate only a *completely observable* version of the system. In this case, we need only consider a base policy that maps underlying states to actions—we could simply apply rollout to this policy, and not have to maintain any belief states in our simulation. Call this method *completely observable (CO) rollout*. It turns out that in certain applications, such as in sensor scheduling for target tracking, a CO-rollout base policy is naturally available (see [22], [23], [27], [40], [41]). Note that we will still need to keep track of (or estimate) the actual belief state of the system, even if we use CO rollout. The benefit of CO rollout is that it allows us to avoid keeping track of (simulated) belief states in our *simulation* of the future evolution of the system.

In designing lookahead methods with a simplified belief state, we must ensure the simplification does not hide the good or bad effects of actions. The resulting Q-value approximation must properly rank current actions. This requires a carefully designed simplification of the belief state together with a base policy that appropriately reflects the effects of taking specific current actions.

For example, suppose that a particular current action results in poor future rewards because it leads to belief states with large variances. Then, if we use the method of CO rollout, we have to be careful to ensure that this detrimental effect of the particular current action be reflected as a cost in the lookahead. (Otherwise, the effect would not be accounted for properly, because in CO rollout we do not keep track of belief states in our simulation of the future effect of current actions.)

Another caveat in the use of simplified belief states in our lookahead is that the resulting rewards in the lookahead may also be affected (and this may have to be taken into account). For example, consider again the problem of sensor scheduling for target tracking, where the per-step reward is the negative mean of the sum of the tracking error and the sensor usage cost. Suppose that we use a particle filter for tracking (i.e., for keeping track of the actual belief state). However, for our lookahead, we use a Kalman filter to keep track of future belief states in our rollout simulation. In general, the tracking error associated with the Kalman filter is different from that of the particle filter. Therefore, when summed with the sensor usage cost, the relative contribution of the tracking error to the overall reward will be different for the Kalman filter compared to the particle filter. To account for this, we will need to scale the tracking error (or sensor usage cost) in our simulation so that the effect of current actions are properly reflected in the Q-value approximations from the rollout with the simplified belief state calculation.

L. *Reward Surrogation*

In applying a POMDP approximation method, it is often useful to substitute the reward function for an alternative (*surrogate*), for a number of reasons. First, we may have a surrogate reward that is much simpler (or more reliable) to calculate than the actual reward (e.g., the method of reduction to classification [8], [9]). Second, it may be desirable to have a single surrogate reward for a range of different actual rewards. For example, [34], [24] shows that average Rényi information gain can be interpreted as a near universal proxy for any bounded performance metric. Third, reward surrogation may be necessitated by the use of a belief-state simplification technique. For example, if we use a Kalman filter to update the mean and covariance of the belief state, then the reward can only be calculated using these entities.

The use of a surrogate reward can lead to many benefits. But some care must be taken in the design of a suitable surrogate reward. Most important is that the surrogate reward be sufficiently reflective of the true reward that the ranking of actions with respect to the approximate Q-values be preserved. A superficially benign substitution may in fact have unanticipated but significant impact on the ranking of actions. For example, recall the example raised in the previous section on belief-state simplification, where we substitute the tracking error of a particle filter for the tracking error of a Kalman filter. Superficially, this substitute appears to be hardly a "surrogate" at all. However, as pointed out before, the tracking error of the Kalman filter may be significantly different in magnitude from that of a particle filter.

VII. Illustration: Spatially Adaptive Airborne Sensing

In this section, we illustrate the performance of several of the strategies discussed in this paper on a common model problem. The model problem has been chosen to have the characteristics of the motivating example given earlier, while remaining simple enough so that the workings of each method are transparent.

In the model problem, there are two targets, each of which is described by a one-dimensional position (see Figure 9). The state is therefore a 2-dimensional real number describing the target locations plus the sensor position, as described in Section III-H. Targets move according to a pure diffusion model (given explicitly in Section III-H as $T_{\text{single target}}(y|x)$), and the belief state is propagated using this model. Computationally, the belief state is estimated by a multi-target particle filter, according to the algorithm given in [35].

The sensor may measure any one of 16 cells, which span the possible target locations (again, see Figure 9). The sensor is capable of making three (not necessarily distinct) measurements per time step, receiving binary returns independent from dwell to dwell. The three measurements are fused sequentially: after each measurement, we update the belief state by incorporating the measurement using Bayes' rule, as discussed in Section III-B. In occupied cells, a detection is received with probability $P_d = 0.9$. In cells that are unoccupied a detection is received with probability P_f (set here at 0.01). This sensor model is given explicitly in Section III-H by $P_{\rm obs}(z|x,a)$.

At the onset, positions of the targets are known only probabilistically. The belief state for the first target is uniform across sensor cells $\{2,\dots,6\}$ and for the second target is uniform across sensor cells $\{11,\dots,15\}$. The particle filter used to estimate the belief state is initialized with this uncertainty.

Visibility of the cells changes with time as in the motivating example of Section III-H. At time 1, all cells are visible. At times 2, 3, and 4, cells $\{11,\dots,15\}$ become obscured. At time 5, all cells are visible again. This time varying visibility map is known to the sensor management algorithm and should be exploited to best choose sensing actions.

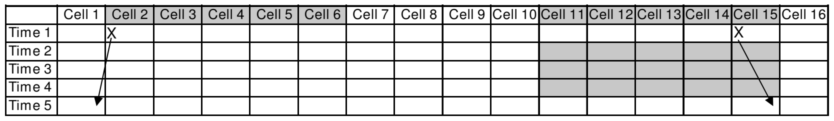


Fig. 9. The model problem. At the onset, the belief state for target 1 is uniformly distributed across cells $\{2,\dots,6\}$ and the belief state for target 2 is uniformly distributed across cells $\{11,\dots,15\}$. At time 1 all cells are visible. At times 2, 3, and 4, cells $\{11,\dots,15\}$ are obscured. This is a simple case where a target is initially visible, becomes obscured, and then reemerges.

Sensor management decisions are made by using the belief state to predict which actions are most valuable. In the following paragraphs, we contrast the decisions made by a number of different strategies that have been described earlier.

At time 1 a myopic strategy, using no information about the future visibility, will choose to measure cells uniformly from the set $\{2,\dots,6\} \cup \{11,\dots,15\}$ as they all have the same expected immediate reward. As a result, target 1 and target 2 will on the average be given equal attention. A nonmyopic strategy, on the other hand, will choose to measure cells from $\{11,\dots,15\}$ as they are soon to become obscured. That is, the policy of looking for target 2 at time 1 followed by looking for target 1 is best.

Figure 10 shows the performance of several of the on-line strategies discussed in this paper on this common model problem. The performance of each scheduling strategy is measured in terms of the mean squared tracking error at each time step. The curves represent averages over $10,000$ realizations of the model problem. Each realization has randomly chosen initial positions of the targets and measurements corrupted by random mistakes as discussed above. The five policies are as follows.

- A **random** policy that simply chooses one of the 16 cells randomly for interrogation. This policy provides a worst-case performance and will bound the performance of the other policies.
- A **myopic** policy that takes the action expected to maximize immediate reward. Here the surrogate reward is myopic information gain as defined in Section VI-D, measured in terms of the expected Rényi divergence with $\alpha = 0.5$ (see [34]). So the value of an action is estimated by the amount of information it gains. The myopic policy is sub-optimal because it does not consider the long term ramifications of its choices. In particular, at time 1 the myopic strategy has no preference as to which target to measure because both are unobscured and have uncertain position. Therefore, half of the time, target 1 is measured, resulting in an opportunity cost because target 2 is about to disappear.
- The **reinforcement learning** approach described in Section VI-F. The Q-function was learned using a linear function approximator, as described in detail in Section VI-F, by running a large number (10^5) of sample vignettes. Each sample vignette proceeds as follows. An action is taken randomly. The resulting immediate gain (as measured by the expected information gain) is recorded and the resulting next-state computed. This next-state is used to predict the long-term gain using the currently available Q-function. The Q-function is then refined given this information (in practice this is done in blocks of many vignettes, but the principle is the same). Training the Q-function is a very time

consuming process. In this case, for each of the 10^5 sample vignettes, the problem was simulated from beginning to end, and the state and reward variables were saved along the way. It is also unclear as to how the performance of the trained Q-function will change if the problem is perturbed. However, with these caveats in mind, once the Q-function has been learned, decision making is very quick and the resulting policy in this case is very good.

- The **heuristic EVTG approximation** described in Section VI-D favors actions expected to be more valuable now than in the future. In particular, actions corresponding to measuring target 2 have additional value because target 2 is predicted to be obscured in the future. This makes the ranking of actions that measure target 2 higher than those that measure target 1. Therefore, this policy (like the other nonmyopic approximations described here) outperforms the myopic policy. The computational burden is on the order of H times the myopic policy, where H is the horizon length.
- The **rollout** policy described in Section VI-H. The base policy used here is to take each of the three measurements sequentially at the location where the target is expected to be, which is a function of the belief state that is current to the particular measurement. This expectation is computed using the predicted future belief state, which requires the belief state to be propagated in time. This is done using a particle filter. We again use information gain as the surrogate reward to approximate Q-values. The computational burden of this method is on the order of NH times that of the myopic policy, where H is the horizon length and N is the number of Monte Carlo trials used in the approximation (here $H = 5$ and $N = 25$).
- The **completely observable rollout** policy described in Section VI-K. As in the rollout policy above, the base policy here is to take measurements sequentially at locations where the target is expected to be, but enforces the criterion that the sensor should alternate looking at the two targets. This slight modification is necessary due to the delta-function representation of future belief states. Since the completely observable policy does not predict the posterior into the future, it is significantly faster than standard rollout (an order of magnitude faster in these simulations). However, it requires a different surrogate reward (one that does not require the posterior like the information gain surrogate metric). Here we have chosen as a surrogate reward to count the number of detections received, discounting multiple detections of the same target.

Our main intent here is simply to convey that, from Figure 10, the nonmyopic policies perform similarly, and are better than the myopic and random policies, though at the cost of additional computational burden. The nonmyopic techniques perform similarly since they ultimately choose similar policies. Each one prioritizes measuring the target that is about to dissapear over the target that is in the clear. On the other hand, the myopic policy is “losing” the target more often, resulting in higher mean error as there are more catastrophic events.

VIII. Illustration: Multi-Mode Adaptive Airborne Sensing

In this section, we turn our attention to adaptive sensing with a waveform-agile sensor. In particular, we investigate how the availability of multiple waveform choices effects the myopic/nonmyopic trade. The model problem considered here again focuses on detection and tracking in a visibility impaired environment. The target dynamics, belief-state update, and observation law are identical to that described in the first simulation. However, in this section we look at a sensor that is agile over waveform as well as pointing direction (i.e., can choose both where to interrogate as well as what waveform to use). Furthermore, the different waveforms are subject to different (time-varying) visibility maps. Simulations show that the addition of waveform agility (and corresponding visibility differences) changes the picture. In this section, we restrict our attention to the EVTG heuristic for approximate nonmyopic planning. Earlier simulations have shown that in model problems of this type, the various approaches presented here perform similarly.

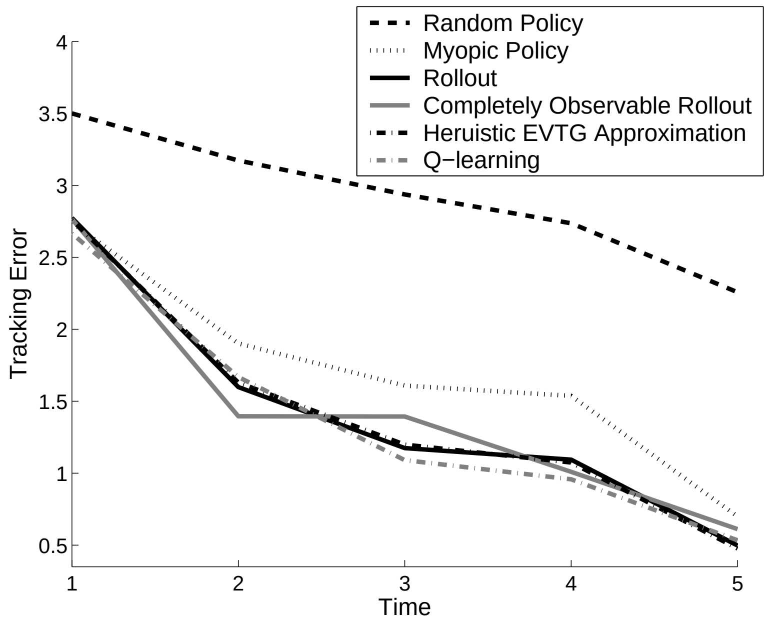


Fig. 10. The performance of the five policies discussed above. Performance is measured in terms of mean squared tracking error at each time step, averaged over a 10^4 Monte Carlo trials.

A. A Study with a Single Waveform

We first present a baseline result comparing random, myopic, and heuristic EVTG (HECTG) approximation based performance in the (modified) model problem. The model problem again covers a surveillance area broken into 16 regions with a target that is to be detected and tracked. The single target moves according to a purely diffusive model, and the belief state is propagated using this model. However, in this simulation the model problem is modified in that there is only one sensor allocation per time step and the detection characteristics are severely degraded. The region is occluded by a time-varying visibility map that obscures certain sub-regions at each time step, degrading sensor effectiveness in those regions at that time step. The visibility map is known exactly *a priori* and can be used both to predict which portions of the region are useless to interrogate at the present time (because of current occlusion) and to predict which regions will be occluded in the future. The sensor management choice in the case of a single waveform is to select the pointing direction (one of the 16 sub-regions) to interrogate. If a target is present and the sub-region is not occluded, the sensor reports a detection with $p_d = 0.5$. If the target is not present or the sub-region is occluded the sensor reports a detection with $p_f = .01$.

Both the myopic and nonmyopic information based methods discount the value of looking at occluded sub-regions. Prediction of myopic information gain uses visibility maps to determine that interrogating an occluded cell provides no information because the outcome is certain (it follows the false alarm distribution). However, the nonmyopic strategy goes further: It uses future visibility maps to predict which sub-regions will be occluded in the future and gives higher priority to their interrogation at present.

The simulation results shown in Figure 11 indicate that the HEVTG approximation to the nonmyopic scheduler provides substantial performance improvement with respect to a myopic policy in the single waveform model problem. The gain in performance for the policy that looks ahead is primarily ascribable to the following. It is important to promote interrogation of sub-regions that are about to become occluded over those that will remain visible. If a sub-region is not measured and then becomes occluded, the opportunity to determine target presence in that region is lost until the region becomes visible again. This opportunity cost is captured in the HEVTG approximation as it predicts which actions will have less value in the future and promotes them at the present. The myopic policy merely looks at the current situation and takes the action with maximal immediate gain. As a result of this greediness, it misses opportunities

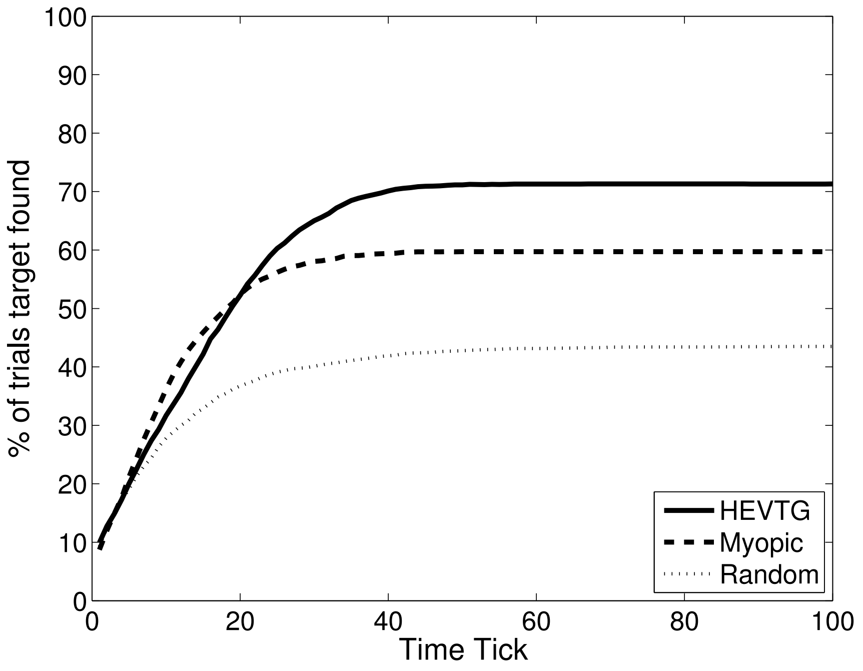


Fig. 11. Performance of the scheduling policies with a pointing-agile single waveform sensor.

that have long term benefit. As a result of this greediness, the myopic policy may outperform the HEVTG in the short term but ultimately underperforms.

B. A Study with Multiple Independent Waveforms

This subsection explores the effect of multiple waveforms on the nonmyopic/myopic trade. We consider multiple *independent* waveforms, where independent means the time-varying visibility maps for the different waveforms are not coupled in any way. This assumption is relaxed in the following subsection.

Each waveform has an associated time-varying visibility map drawn independently from the others. The sensor management problem is one of selecting both pointing direction and the waveform. All other simulation parameters are set identically to the previous simulation (i.e., detection and false alarm probabilities, and target kinematics). Figure 12 shows performance curves for two and five independent waveforms. In comparison to the single waveform simulation, these simulations (a) have improved overall performance, and (b) have a narrowed gap in performance between nonmyopic and myopic schedulers.

Figure 13 provides simulation results as the number of waveforms available is varied. These results indicate that as the number of independent waveforms available to the scheduler increase, the performance difference between a myopic policy and a nonmyopic policy narrows. This is largely due to the softened opportunity cost the myopic policy suffers. In the single waveform situation, if a region became occluded it could not be observed until the visibility for the single waveform changed. This puts a sharp penalty on a myopic policy. However, in the multiple independent waveform scenario, the penalty for myopic decision making is much less severe. In particular, if a region becomes occluded in waveform i, it is likely that some other waveform is still viable (i.e., the region is unoccluded to that waveform) and a myopic policy suffers little loss. As the number of independent waveforms available to the sensor increases, this effect is magnified until there is essentially no difference in the two policies.

C. A Study with Multiple Coupled Waveforms

A more realistic multiple waveform scenario is one in which the visibility occlusions between waveforms are highly coupled. Consider the case where a platform may choose between the following 5 waveforms

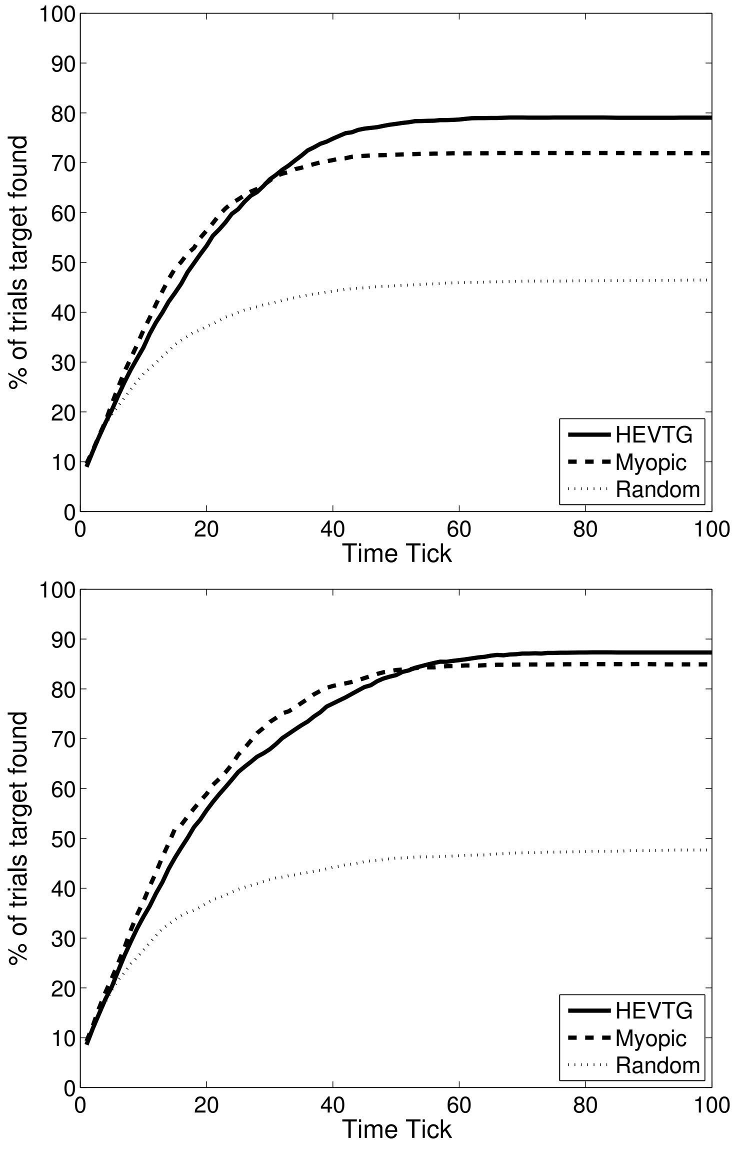


Fig. 12. Top: Performance of the strategies with a two-waveform sensor. Bottom: Performance curves with a five-waveform sensor.

(modalities) for interrogation of a region: electro-optical (EO), infra-red (IR), synthetic aperture radar (SAR), foliage penetrating radar (FOPEN), and moving target indication radar (MTI). In this situation, the visibility maps for the 5 waveforms are highly coupled through the environmental conditions (ECs) present in the region. For example, clouds effect the visibility of both EO and IR. Similarly, tree cover effects the performance of all modes except FOPEN, and so on.

Therefore, a more realistic study of multiple waveform performance is to model the time-varying nature of a collection of environmental conditions and generate the (now coupled) waveform visibility maps from the ECs. For this simulation study, we choose the nominal causation map shown in Figure 14(L).

The time-varying maps of each EC are chosen to resemble a passover, where for example the initial cloud map is chosen randomly and then it moves at a random orientation and random velocity through the region over the simulation time. The waveform visibility maps are then formed by considering all obscuring ECs and choosing the maximum obscuration. This setup results in fewer than five independent waveforms available to the sensor because the viability maps are coupled through the ECs.

Figure 14 (right) shows a simulation result of the performance for a five waveform sensor. The simulation shows the gap between the myopic policy and the nonmyopic policy widens from where it was in the

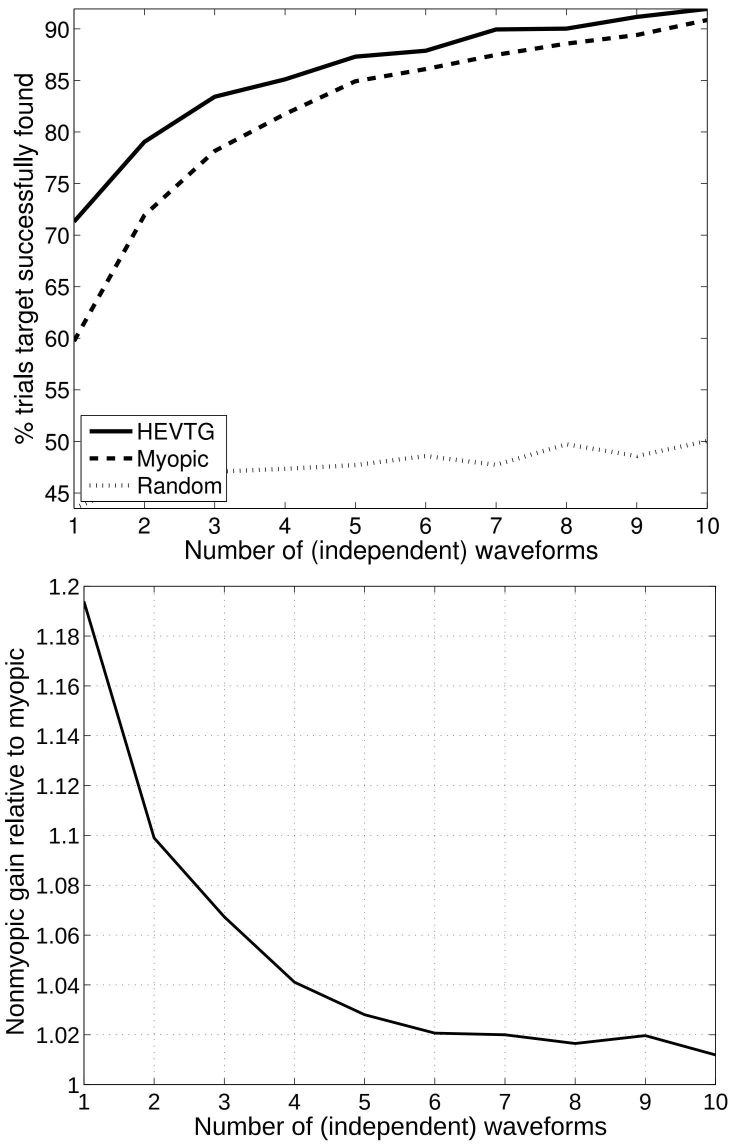


Fig. 13. Top: The terminal performance of the scheduling algorithms versus number of waveforms. Bottom: The gain (performance improvement) of the nonmyopic policy with respect to the myopic policy.

independent waveform simulation. In fact, in this scenario, the 5 dependent waveforms have performance characteristics that are similar to 2 independent waveforms, as measured by the ratio of nonmyopic scheduler performance to myopic scheduler performance. Figure 15 illustrates the difference among the three policies being compared here, highlighting the "lookahead" property of the nonmyopic scheme.

IX. Conclusions

This paper has presented methods for adaptive sensing based on approximations for partially observable Markov decision processes, a special class of discrete event system models. Though we have not specifically highlighted the event-driven nature of these models, our framework is equally applicable to models that are more appropriately viewed as event driven. The methods have been illustrated on the problem of waveform-agile sensing, wherein it has been shown that intelligently selecting waveforms based on past outcomes provides significant benefit over naive methods. We have highlighted, via simulation, computationally approaches based on rollout and a particular heuristic related to information gain. We have detailed some of the design choices that go into finding appropriate approximations, including choice

	Cloud	Rain	Wind	Fog	Foliage
EO	X	X	X	X	X
SAR		X			X
FOPEN		X	X		
IR	X	X		X	X
GMTI		X			X

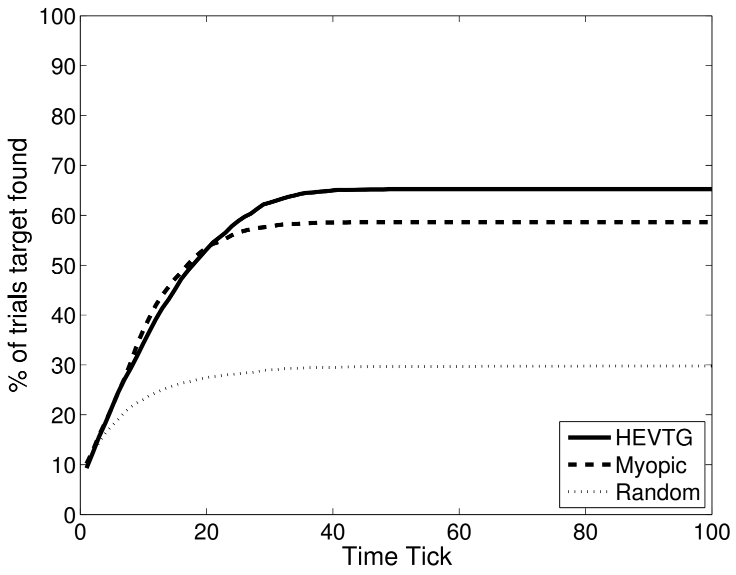


Fig. 14. Top: EC Causation map. Bottom: Performance of the scheduling strategies with a pointing-agile five waveform sensor, where the visibility maps are coupled through the presence of environmental conditions.

of surrogate reward and belief-state representation.

Throughout this paper we have taken special care to emphasize the limitations of the methods. Broadly speaking, all tractable methods require domain knowledge in the design process. Rollout methods require a base policy specially designed for the problem at hand; relaxation methods require one to identify the proper constraint(s) to remove; heuristic approximations require identification of appropriate value-to-go approximations, and so on. That being said, when domain knowledge is available it can often yield dramatic improvement in system performance over traditional methods at a fixed computational cost. Formulating a problem as a POMDP itself poses a number of challenges. For example, it might not be straightforward to cast the optimization objective of the problem into an expected cumulative reward (with stagewise additivity).

A number of extensions to the basic POMDP framework are possible. First, of particular interest to discrete event systems is the possibility of event-driven sensing, where actions are taken only after some event occurs or some condition is met. In this case, the state evolution is more appropriately modeled as a semi-Markov process (though with some manipulation it can be converted into an equivalent standard Markovian model) [55, Ch. 7]. A second extension is to incorporate explicit constraints into the decision-making framework [1], [13], [62].

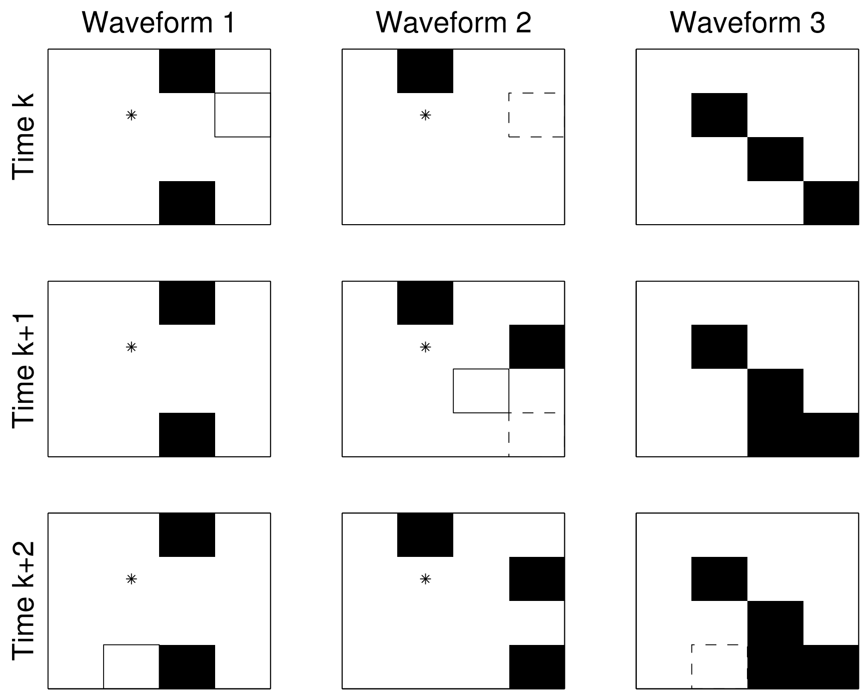


Fig. 15. Three time steps from a three waveform simulation. Obscured areas are shown with filled black squares and unobscured areas are white. The true target position is shown by an asterisk for reference. The decisions (waveform choice and pointing direction) are shown with solid-bordered squares (myopic policy) and dashed-bordered squares (nonmyopic policy). This illustrates "lookahead," where regions that are about to be obscured are measured preferentially by the nonmyopic policy.

References

[1] E. Altman, *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1998.

[2] R. Bartels, S. Backus, E. Zeek, L. Misoguti, G. Vdovin, I. P. Christov, M. M. Murnane, and H. C. Kapteyn, "Shaped-pulse optimization of coherent soft X-rays," *Nature*, vol. 406, pp. 164–166, 2000.

[3] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[5] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," in *Proc. Joint 44th IEEE Conf. on Decision and Control and European Control Conf.*, Seville, Spain, December 12–15, 2005.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, Vol. I, 3rd Ed., 2005; Vol. II, 3rd Ed., 2007.

[7] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, pp. 89–108, 1999.

[8] D. Blatt and A. O. Hero III, "From weighted classification to policy search," in *Advances in Neural Information Processing Systems (NIPS) 18*, 2006, pp. 139–146.

[9] D. Blatt and A. O. Hero III, "Optimal sensor scheduling via classification reduction of policy search (CROPS)," in *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2006.

[10] D. Castanon,"Approximate dynamic programming for sensor management," *Proc. 36th IEEE Conf. on Decision and Control*, San Diego, December 1997, pp. 1202–1207.

[11] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-based Algorithms for Markov Decision Processes*. Springer series in Communications and Control Engineering, 2007.

[12] H. S. Chang, R. L. Givan, and E. K. P. Chong, "Parallel rollout for online solution of partially observable Markov decision processes," *Discrete Event Dynamic Systems*, vol. 14, no. 3, pp. 309–341, 2004.

[13] R. C. Chen and K. Wagner, "Constrained partially observed Markov decision processes for adaptive waveform scheduling," in *Proc. Int. Conf. on Electromagnetics in Advanced Applications*, Torino, September 17–21, 2007, pp. 454–463.

[14] H. T. Cheng, *Algorithms for Partially Observable Markov Decision Processes*. PhD dissertation, University of British Columbia, 1988.

[15] A. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "Efficient search strategies for non-myopic sensor scheduling in target tracking," *Asilomar Conf. on Signals, Systems, and Computers*, November 2004.

[16] E. Çinlar, *Introduction to Stochastic Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[17] E. K. P. Chong, R. L. Givan, and H. S. Chang, "A framework for simulation-based network control via hindsight optimization," *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, Dec. 12–15, 2000, pp. 1433–1438.

[18] D. P. de Farias and B. Van Roy. "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[19] D. P. de Farias and B. Van Roy, “On constraint sampling in the linear programming approach to approximate dynamic programming,” *Mathematics of Operations Research*, vol. 29, no. 3, pp. 462–478, August 2004.

[20] E. Gottlieb and R. Harrigan, “The Umbra Simulation Framework,” Sandia Tech. Report SAND2001-1533 (Unlimited Release), June 2001.

[21] J. A. Gubner, *Probability and Random Processes for Electrical and Computer Engineers*. New York, NY: Cambridge University Press, 2006.

[22] Y. He and E. K. P. Chong, “Sensor scheduling for target tracking in sensor networks,” in *Proc. 43rd IEEE Conf. on Decision and Control (CDC’04)*, December 14–17, 2004, pp. 743–748.

[23] Y. He and E. K. P. Chong, “Sensor scheduling for target tracking: A Monte Carlo sampling approach,” *Digital Signal Processing*, vol. 16, no. 5, pp. 533–545, September 2006.

[24] A. Hero, D. Castanon, D. Cochran, and K. Kastella, Eds., *Foundations and Applications of Sensor Management*, Springer, 2008.

[25] S. Ji, R. Parr, and L. Carin, “Nonmyopic multiaspect sensing with partially observable Markov decision processes,” *IEEE Trans. Signal Processing*, vol. 55, no. 6, Part 1, pp. 2720–2730, June 2007.

[26] S. Julier and J. Uhlmann, “Unscented filtering and nonlinear estimation,” *Proc. IEEE*, vol. 92, no. 3, pp. 401–422, March 2004.

[27] L. W. Krakow, Y. Li, E. K. P. Chong, K. N. Groom, J. Harrington, and B. Rigdon, “Control of perimeter surveillance wireless sensor networks via partially observable Markov decision process,” in *Proc. 2006 IEEE Int. Carnahan Conf. on Security Technology (ICCST)*, Lexington, Kentucky, October 17–20, 2006.

[28] M. J. Kearns, Y. Mansour, and A. Y. Ng, “A sparse sampling algorithm for near-optimal planning in large Markov decision processes,” *Proc. 16th Int. Joint Conf. on Artificial Intelligence*, 1999, pp. 1324–1331.

[29] L. P. Kaelbling, M. L. Littman and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[30] L. P. Kaelbling, M. L. Littman and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[31] C. M. Kreucher, D. Blatt, A. O. Hero III, and K. Kastella, “Adaptive multi-modality sensor scheduling for detection and tracking of smart targets,” *Digital Signal Processing*, 2006.

[32] C. M. Kreucher, A. O. Hero, K. Kastella, and D. Chang, “Efficient methods of non-myopic sensor management for multitarget tracking,” in *Proc. 43rd IEEE Conf. on Decision and Control (CDC’04)*, December 14–17, 2004.

[33] C. M. Kreucher, A. Hero, and K. Kastella, “A Comparison of task driven and information driven sensor management for target tracking,” in *Proc. 44th IEEE Conf. on Decision and Control (CDC’05)*, December 12–15, 2005.

[34] C. M. Kreucher, K. Kastella, and A. O. Hero III, “Sensor management using an active sensing approach,” *Signal Processing*, vol. 85, no. 3, pp. 607–624, March 2005.

[35] C. M. Kreucher, K. Kastella, and A. O. Hero III, “Multitarget tracking using the joint multitarget probability density,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1396–1414, October 2005.

[36] V. Krishnamurthy and R. J. Evans, “Hidden Markov model multiarm bandits: A methodology for beam scheduling in multitarget tracking,” *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 2893–2908, December 2001.

[37] V. Krishnamurthy, “Emission management for low probability intercept sensors in network centric warfare,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 1, pp. 133–151, January 2005.

[38] W. S. Lovejoy, “Computationally feasible bounds for partially observed Markov decision processes,” *Operations Research*, vol. 39, pp. 162–175, 1991.

[39] W. S. Lovejoy, “A survey of algorithmic methods for partially observed Markov decision processes,” *Annals of Operations Research*, vol. 28, no. 1, pp. 47–65, December 1991.

[40] Y. Li, L. W. Krakow, E. K. P. Chong, and K. N. Groom, “Dynamic sensor management for multisensor multitarget tracking,” *Proc. 40th Annual Conf. on Information Sciences and Systems*, Princeton, New Jersey, March 22–24, 2006 pp. 1397–1402.

[41] Y. Li, L. W. Krakow, E. K. P. Chong, and K. N. Groom, “Approximate stochastic dynamic programming for sensor scheduling to track multiple targets,” *Digital Signal Processing*, 2007, in press. doi:10.1016/j.dsp.2007.05.004

[42] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. London: Springer-Verlag, 1993.

[43] S. A. Miller, Z. A. Harris, and E. K. P. Chong, “A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking,” *EURASIP Journal on Applied Signal Processing*, special issue on *Signal Processing Advances in Robots and Autonomy*, in press, to appear 2009.

[44] L. S. Pontryagin, V.G. Boltyansky, R.V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*. New York: Wiley, 1962.

[45] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: Wiley-Interscience, 2007.

[46] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Norwood, MA: Artech House, 2004.

[47] S. M. Ross, *Applied Probability Models with Optimization Applications*. New York, NY: Dover Publications, 1970.

[48] N. Roy, G. Gordon, and S. Thrun, “Finding approximate POMDP solutions through belief compression,” *Journal of Artificial Intelligence Research*, vol. 23, pp. 1–40, 2005.

[49] J. Rust, “Using randomization to break the curse of dimensionality,” *Econometrica*, vo. 65, no. 3, pp. 487–516, May 1997.

[50] W. R. Scott, Jr., K. Kim, G. D. Larson, A. C. Gurbuz, and J. H. McClellan, “Combined seismic, radar, and induction sensor for landmine detection,” in *Proc. 2004 Int. IEEE Geoscience and Remote Sensing Symposium*, Anchorage, Alaska, September 20–24, 2004, pp. 1613–1616.

[51] L. Shi and C.-H. Chen, “A new algorithm for stochastic discrete resource allocation optimization,” *Discrete Event Dynamic Systems*, vol. 10, pp. 271–294, 2000.

[52] R. D. Smallwood and E. J. Sondik, “The optimal control of partially observable Markov processes over a finite horizon,” *Operations Research*, vol. 21, pp. 1071–1088, no. 5, 1973.

[53] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, MIT Press, 1998.

[54] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*. Cambridge, MA: The MIT Press, 2005.

[55] H. C. Tijms, *A First Course in Stochastic Models*. New York: Wiley, 2003.

[56] R. Washburn, M. Schneider, and J. Fox. "Stochastic dynamic programming based approaches to sensor resource management," *5th Int. Conf. on Information Fusion*, 2002.

[57] C. J. C. H. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, King's College, University of Cambridge, England 1989.

[58] J. C. Willems, "1969: The birth of optimal control," in *Proc. 35th IEEE Conf. on Decision and Control (CDC'96)*, December 1996, pp. 1586–1587.

[59] G. Wu, E. K. P. Chong, and R. L. Givan, "Burst-level congestion control using hindsight optimization," *IEEE Transactions on Automatic Control*, special issue on *Systems and Control Methods for Communication Networks*, vol. 47, no. 6, pp. 979–991, June 2002.

[60] H. Yu and D. P. Bertsekas, "Discretized approximations for POMDP with average cost," in *Proc. 20th Conf. on Uncertainty in Artificial Intelligence*, Banff, Canada, 2004, pp. 619–627.

[61] N. L. Zhang and W. Liu, *Planning in Stochastic Domains: Problem Characteristics and Approximation*, Tech. Report HKUST-CS96-31, Dept. of Computer Science, Hong Kong University of Science and Technology, 1996.

[62] Z. Zhang, S. Moola, and E. K. P. Chong, "Approximate stochastic dynamic programming for opportunistic fair scheduling in wireless networks," in *Proc. 47th IEEE Conf. on Decision and Control*, Cancun, Mexico, December 9–11, 2008, pp. 1404–1409.