Abstract—In this paper we study the problem of tracking an object moving randomly through a network of wireless sensors in the presence of clutter. Our objective is to devise strategies for scheduling the sensors to optimize the tradeoff between tracking performance and energy consumption. The presence of random interference introduces uncertainty into the origin of the measurements. Data association techniques are thus required to associate each measurement with the target or discard it as arising from clutter (false alarms). We cast the scheduling problem as a Partially Observable Markov Decision Process (POMDP), where the control actions correspond to the set of sensors to activate at each time step. Exact solutions are generally intractable even for the simplest models due to the dimensionality of the information and action spaces. Hence, we develop an approximate sensor scheduler that optimizes a point-based value function over a set of reachable beliefs. Point-based updates are driven by a non-linear filter that combines the validated measurements through proper association probabilities.

Our approach efficiently combines Probabilistic Data Association techniques for belief update with Point-Based Value Iteration for designing scheduling policies. The generated scheduling policies, albeit suboptimal, provide good energy-tracking tradeoffs.

I. INTRODUCTION

Smart sensor management holds the potential to optimize the usage of sensor networks which typically operate on limited resources. A significant body of research work considers tasking sensors in dynamically evolving environments for a wide range of applications including tracking, classification, monitoring and surveillance applications [1], [7], [9], [11], [13], [20]. This paper focuses on sensor scheduling for tracking in cluttered environments. Our goal is to design a central controller to schedule the sensors to track an object of interest in the presence of false alarms (clutter).

A filtering component (linear or non-linear) is part of any sensor management algorithm, not only for the obvious purpose of state estimation, but also to provide the statistics necessary for the controller to select appropriate sensing actions and modes. Non-linear filtering for tracking in cluttered environments is particularly hard as it requires considering a large number of events due to the so-called data association problem, and is hence computationally intensive. The presence of random interference from nearby objects, false alarms, electromagnetic interference etc. generally leads to ambiguity in the origin of the sensor measurements and hence it is crucial to associate the measurements with their corresponding tracks. One simple and intuitive candidate solution for the association problem is to choose the signal with the highest intensity, among a set of validated measurements, for track update and discard the others. This is known as Strongest Neighbor Filter SNF [5]. The Nearest Neighbor Filter NNF is another solution that uses the measurement closest to the predicted measurement obtained through a prediction step of the track estimation filter [2]. However, these algorithms start to fail when the false alarm rate, or clutter density, increases. Alternatively, probabilistic data association (PDA) for a single target in clutter is another approach which uses all the validated measurements and does not discard any of them [3]. A proper weight, reflecting the association probability, is assigned to each measurement and the weighted average of the validated innovations is used for the update.

While most of the existing literature on target tracking in clutter has focused on the estimation aspect of the tracking problem using one or two sensors, the primary focus of this paper is on the design of efficient control policies organizing the activity of a larger network of sensors in the presence of false alarms. We cast the scheduling problem as a Partially Observable Markov Decision Process (POMDP), and devise strategies whereby the sensors are activated to optimize the fundamental tradeoff between energy expenditure and tracking performance in the presence of spurious measurements from clutter.

Solving a POMDP optimally (to find the optimal control policy) is generally intractable. For example, the value function for a POMDP with a finite state space depends on information states consisting of conditional probability vectors (beliefs) of dimension equal to the number of states. This has led to a number of POMDP approximations and we refer the reader to Monahan [14] and Hauskrecht [10] for excellent surveys on approximate methods for stochastic dynamic programming. Our approach combines PDA for belief update with approximate Point-Based Value Iteration for designing scheduling policies. A PDA filter is used to construct beliefs about the state of the system for state estimation and for the purpose of providing the controller with sufficient statistics for control design.

The rest of the paper is organized as follows. In section II we describe the sensor scheduling problem and the belief update for different sensing models. In Section III we first provide
a brief overview of approximate solutions for POMDPs then present our proposed point-based scheduler. Examples are then considered and simulation results are presented in Section IV. Finally, Section V provides concluding remarks.

II. PROBLEM SETUP

A. A simple sensing model

We focus in this paper on the design of scheduling policies rather than the tracking aspect of the problem. Hence, we first consider a simplistic model for sensing wherein the network is divided into \( n \) distinct cells, one for each sensor. Therefore, the sensing ranges of different sensors do not overlap as each cell corresponds to the sensing range of one particular sensor. This assumption is relaxed when we consider sensors with overlapping sensing regions in Section II-B.

The movement of the object is described by a Markov chain with an \((n+1) \times (n+1)\) probability transition matrix \( P \). The extra state is an absorbing termination state \( \tau \) accounting for the case where the object leaves the network. We further assume that \( \tau \) is perfectly observable, i.e., when the object leaves the network the problem terminates. Information about the object trajectory is stored at some central unit which keeps track of the system’s state and determines the scheduling actions for the different sensors. We let \( b_k \) denote the location of the tracked object at time \( k \). The action for sensor \( \ell \) at time \( k \) is denoted \( u_{k,\ell} \) such that \( u_{k,\ell} = 1 \) if sensor \( \ell \) is activated at time \( k+1 \) and 0 otherwise. The action vector at time \( k \), denoted \( u_k \), is a binary vector of size \( n \times 1 \), one decision per sensor. In this simplistic model, we assume that the target is perfectly observable to an awake sensor within its sensing range or if it reaches the terminal state \( \tau \), otherwise it is unobservable. However, active sensors could falsely declare the presence of a target within their cell due to random interference (clutter). Thus, if \( s_k \) is the measurement vector at time \( k \) and \( s_{k,\ell} \) the observation of the \( \ell \)-th sensor, then

\[
s_{k, b_k} = \begin{cases} 
1, & \text{if } u_{k-1, b_k} = 1; \\
0, & \text{if } u_{k-1, b_k} = 0;
\end{cases}
\]

and

\[
s_{k, \ell} = \begin{cases} 
1, \quad \text{w.p. } P_F & \text{if } u_{k-1, \ell} = 1; \\
0, \quad \text{if } u_{k-1, \ell} = 0; \\
0, \quad \text{w.p. } 1 - P_F & \text{if } u_{k-1, \ell} = 1; \\
& \forall \ell \neq b_k
\end{cases}
\]

where w.p. is a shorthand for “with probability”. The clutter density is captured by the false alarm probability \( P_F \) that an active sensor provides a positive measurement. Therefore, clutter leads to uncertainty into the origin of the measurements which could eventually lead to loss in tracking performance. Proper countermeasures should take that into consideration when designing a sensor scheduling policy.

B. Overlapping sensing regions

In this model, we allow the sensing regions to overlap. An example of this model is illustrated in Fig.1 depicting a network of \( n = 12 \) sensors observing \( m = 20 \) potential object locations according to the shown connectivity. If \( B_{b_k} \) is the set of sensors observing the target at time \( k \), then the observation model of the \( \ell \)-th sensor is given by

\[
p(s_{k,\ell} = 1| b_k, u_{k-1,\ell} = 1) = \begin{cases} 
1, & \text{if } \ell \in B_{b_k}; \\
P_F, & \text{if } \ell \notin B_{b_k}
\end{cases}
\]

That is, when the target is in the vicinity of an active sensor, the sensor gets a positive observation, however, active sensors which do not belong to the set \( B_{b_k} \) could also falsely declare a target present with probability \( P_F \). This discrete model is simplistic, yet it captures essential features in real sensing systems, namely, overlapping sensing ranges, limited visibility for each sensor, as well as geographical neighborhood properties.

C. Optimal scheduling policy

The design of an optimal scheduling policy depends on the history up to time \( k \), i.e., the information state \( I_k \), which consists of the initial belief \( p_0 \) about the system state, the previous observations, and the previous scheduling actions

\[
I_k = \{ p_0, s_0, s_1, \ldots, s_k, u_0, u_1, u_{k-1} \}
\]

However, the posterior probability distribution of the system’s state given all the history, \( p_k = \Pr[b_k|I_k] \), is a sufficient statistic for this class of partially observable processes [4]. The distribution \( p_k \), also known as belief, summarizes all the information needed for optimal control. It is a \( 1 \times m \) row vector, where \( m \) is the size of the state space with \( m = n + 1 \) for the simple sensing model in Section II-A. The sufficient statistic itself forms a Markov process whose evolution can be obtained through Bayes’ rule updates:

\[
p_{k+1} = \phi(p_k, s_{k+1}, u_k)
\]

where \( \phi(.) \) is a transformation mapping the current belief \( p_k \), the current control vector \( u_k \), and the future observation \( s_{k+1} \), to a future belief \( p_{k+1} \).

The scheduling policy \( u_k = \mu_k(I_k) \) is defined as a mapping from information states \( I_k \) to control actions \( u_k \).
At each time step, the incurred cost $g$ is the sum of the energy and the tracking costs. An energy cost of $c \in (0, 1]$ per unit time is incurred for every active sensor and a tracking cost of 1 for each time unit that the object’s state estimate $\hat{b}_k = \xi_k(I_k)$ is not equal to the true state $b_k$. Once state $\tau$ is reached the problem terminates and no further cost is incurred. In other words, $\tau$ is an absorbing cost-free state. Since $p_k$ is a sufficient statistic for optimal control, then the overall control at time $k$, which consists of both the state estimate and the scheduling control, is

$$\hat{b}_k = \mu_k(p_k) = (\xi_k(p_k), \mu_k(p_k)) = (\hat{b}_k, u_k)$$

Hence,

$$g(b_k, \hat{b}_k) = I(b_k \neq \tau) \left( I(b_k \neq b_k) + \sum_{\ell=1}^n c I(\ell, \ell+1) \right)$$

where $I(\cdot)$ is the indicator function. The parameter $c$ is thus used to tradeoff energy consumption and tracking errors. Note that even though $\hat{b}_k$ is part of the control input, we did not include its past values in $I_k$ since it does not affect the state evolution. Furthermore, as it only affects the cost, the optimal choice of $\hat{b}_k$ would have to minimize the tracking error over a single time step, i.e.,

$$\hat{b}_k = \arg \max_b p_k(b)$$

The goal is to design a policy that minimizes the expected sum of costs $J$, where,

$$J(I_0, \hat{\mu}_0, \hat{\mu}_1, \ldots) = \mathbb{E} \left[ \sum_{k=0}^{\infty} g(b_k, \hat{b}_k) \right] | I_0 , \hat{\mu}_0, \hat{\mu}_1, \ldots.$$  \hspace{1cm} (6)

The function $J$ is well-defined since $g$ is upper bounded by $cn+1$ and the expected time till the object exits the network is finite. Note that the termination is inevitable, thus the objective is to reach the termination state with minimal expected cost. Hence, the optimal policy is the solution to the minimization problem,

$$J^* = \min_{\hat{\mu}_0, \hat{\mu}_1, \ldots} J(I_0, \hat{\mu}_0, \hat{\mu}_1, \ldots)$$

This POMDP problem falls within the class of infinite horizon stochastic shortest path problems. Noting that the termination state is observable, cost-free and absorbing, and that every policy is proper\footnote{A proper policy is a policy that leads to the termination state with probability one regardless of the initial state. In our problem, the scheduling policy does not affect the target motion and all policies are proper in the sense that there is a positive probability that the target will reach the termination state after a finite number of stages.}, a stationary scheduling policy $\mu_\ast(\cdot)$, i.e., one which does not depend on $k$, is optimal in the class of all history-dependent policies and $p_k$ is a sufficient statistic for control \[4\], i.e., $u_k = \mu_\ast(p_k)$, is defined through a time-invariant mapping from the belief space to the action space. The value function $J$ can be written in terms of the sufficient statistic and the optimal policy can be obtained from the solution of the Bellman equation:

$$J(p) = \min_{\hat{u} = (b, u)} E[g(b, \hat{u}) | p, \hat{u}] + \sum_s p(s | p, u) J(\phi(p, s, u))$$

such that $J(e_\tau) = 0$, where $e_i$ is a row vector with a 1 at the $i$-th entry and 0 elsewhere. Note that we removed the time dependence due to the aforementioned time invariance property.

D. Belief Update and Data Association

Since the optimal policy depends on the posterior distribution of the state given all the past, we have to keep track of the belief. If we consider the simple model in Section II-A in the absence of clutter, then the belief update equation at time $k$ simply becomes

$$p_k = \begin{cases} e_\tau, & \text{if } s_k = \tau; \\ e_{b_k}, & \text{if } u_{k-1} = 1; \\ [p_{k-1}]_{j:u_{k-1,j} = 0}, & \text{if } u_{k-1} = 0. \end{cases}$$

The vector $[p_{k-1}]_S$ is the probability vector formed by setting the $i$-th entry $[p_{k-1}]_i$ of the vector $p_{k-1}P$ to zero, $\forall i \notin S$, and then normalizing the vector into a probability distribution. The set $\{ j : u_{k-1,j} = 0 \}$ signifies the set of deactivated sensors. In other words, the updated belief for the model in II-A, is a point mass distribution concentrated at $\tau$ if the object exits the network, and concentrated at $b_k$ if the object is observed. When the object is unobservable, we eliminate the probability mass at all sensors that are awake, since the object cannot be at these locations, and normalize.

However, in the presence of clutter, the estimation problem becomes more involved. We have to adapt our filter to account for the uncertainty in the origin of sensor measurements. We let $\mathcal{A}(k)$ denote the set of active sensors declaring a target at time $k$, and $\mathcal{A}_i(k)$, $i = 1, \ldots, |\mathcal{A}(k)|$, its $i$-th element.

Now define the events

$$\theta_i(k) \triangleq \{ s_{k, \mathcal{A}(k)} \text{ is target originated} \}, \ i = 1, \ldots, |\mathcal{A}(k)|$$

and

$$\theta_0(k) \triangleq \bigcap_i \theta_i^c(k)$$

with probabilities

$$\beta_i(k) = \Pr(\theta_i(k) | I_k), \ i = 0, \ldots, |\mathcal{A}(k)| + 1 \hspace{1cm} (12)$$

where $\theta_i^c(k)$ denotes the complement of the event $\theta_i(k)$. Therefore, $\theta_0(k)$ refers to the event where none of the measurements at time $k$ is target-originated.

Using the total probability theorem, the new belief at time $k$ can be written as

$$p_k = p_k(b_k | I_k) = \sum_{\theta_i} p(b_k | I_k, \theta_i(k)) p(\theta_i(k) | I_k)$$

\hspace{1cm} (13)
Again consider the sensing model in Section II-A. Given the observation model in (1) and (2)
\[
P(b_k | I_k, \theta_i) = \begin{cases} 
  e_{A_i}(k), & \forall i \notin \{0, |A(k)| + 1\}; \\
  e_{\tau}, & \text{if } i = |A(k)| + 1; \\
  [p_{k-1}P]_{\{j: u_{k-1,j} = 0\}}, & \text{if } i = 0
\end{cases}
\]
(14)

In other words, conditioned on the association event \(\theta_0\) we eliminate the probability mass at all sensors that are awake, since the object cannot be at these locations, and normalize. Hence,
\[
P_k = \beta_0(k)[p_{k-1}P]_{\{j: u_{k-1,j} = 0\}} + \sum_{i=1}^{|A(k)|} \beta_i(k)e_{A_i}(k)
+ \beta_{|A(k)|+1}(k)e_{\tau}
\]
(15)

Next we compute the association probabilities \(\beta_i(k) = P(\theta_i|I_k)\). Using Bayes’ rule
\[
\beta_i(k) = P(\theta_i|I_{k-1}, u_{k-1}, s_k) \\
\propto p(s_k|\theta_i, I_{k-1}, u_{k-1})P(\theta_i|I_{k-1}) \\
= p(s_k|\theta_i, I_{k-1}, u_{k-1}) \sum_{b_{k-1}} P(\theta_i|b_{k-1}, I_{k-1})P(b_{k-1}|I_{k-1})
\]
(16)

Hence
\[
\beta_i(k) = F_{|A(k)|-1}^{|A(k)|} \prod_{j \in A(k)} \delta(s_{k,j} - 1) \prod_{\ell \notin A(k)} \delta(s_{k,\ell})
\times \sum_{b_{k-1}} Pr(b_k = A_i(k)|b_{k-1})p_{k-1}, \ i = 1, \ldots, |A(k)|
\]
(17)

and
\[
\beta_0(k) = F_{|A(k)|}^{|A(k)|} \prod_{j \in A(k)} \delta(s_{k,j} - 1) \prod_{\ell \notin A(k)} \delta(s_{k,\ell})
\times [p_{k-1}P]_{\{j: u_{k-1,j} = 0\}}
\]
(18)

An analogous approach can be used to write the filtering equations for the overlap model in Section II-B but the evolution is generally more difficult to write mathematically in a compact form. Procedurally, it follows the exact same approach described above. All the hypotheses are first enumerated. Then, the evolution is obtained as a weighted combination of the evolution under each individual hypothesis where the weights correspond to the association probabilities as in (13). However, the number of hypotheses in this case is significantly larger since under one hypothesis it might very well be the case that multiple measurements from awake sensors are target originated because the sensing regions of different sensors overlap. The number of hypotheses generally scales exponentially with the number of active sensors. To reduce the space of association hypotheses, the controller could limit the maximum number of sensors to be activated at any time step to a relatively small number, say \(n_1\) sensors. As we will show in our simulations results, we choose \(n_1 = 5\), i.e., at most 5 sensors could be active at a given time instant.

III. APPROXIMATE SOLUTION

In this section, we present our proposed point-based schedul er which approximates the optimal solution to (8) using a point-based approximation driven by the non-linear filters described previously in Section II-D. First, we provide a quick overview of the structure of a POMDP and the associated challenges.

A. Background: Solving a POMDP

There are a number of algorithms for solving POMDPs exactly [8], [12], [18]. These algorithms rely on the powerful result of Sondik that the optimal value function for any POMDP can be approximated arbitrarily closely using a set of hyper-planes (\(\alpha\)-vectors) defined over the belief simplex [18]. This fact is the basis for exact value iteration based algorithms, such as the Witness algorithm [6] for computing the value function. The result is a value function parameterized by a number of hyper-planes (or vectors) whereby the belief space is partitioned into a finite number of regions. Each vector minimizes the value function over a certain region of the belief space and has a control action associated with it, which is the optimal control for beliefs in its region.

To clarify, in value iteration we generally start with some initial estimate for \(J^*\) and repeatedly apply the transformation defined by the right hand side of Bellman equation (8) until the sequence of cost functions converges. Let \(\{\alpha^{(k)}_i\}_{i=1}^{J(k)}\) denote the set of vectors parameterizing the value function \(J^{(k)}\) after \(k\) iterations, where \(|J^{(k)}|\) is the total number of hyper-planes, and \(\alpha^{(k)}_i(b)\), which is a hyperplane in the belief space, represents the value of executing the \(k\)-step policy associated with the \(i\)-th vector starting from a state \(b\). Hence, the value of executing the \(i\)-th hyperplane policy starting from a belief state \(p\) is simply the dot product of \(\alpha^{(k)}_i\) and \(p\):
\[
J^{(k)}_i(p) = \sum_b p(b)\alpha^{(k)}_i(b) = p \cdot \alpha^{(k)}_i
\]
Therefore, the value of the optimal \(k\)-step policy starting at \(p\) is simply the minimum dot product over all hyperplanes, i.e.,
\[
J^{(*)}(p) = \min_{\{\alpha^{(k)}_i\}} p \cdot \alpha^{(k)}_i
\]
Hence, \(J^{(*)}(p)\) is piecewise linear and concave. Some of the vectors (also known as policy trees) may be dominated by others in the sense that they are not optimal at any region in the belief simplex. Thus, many exact algorithms devise pruning mechanisms whereby a parsimonious representation with a minimal set of non-dominated hyper-planes is maintained [14].

B. Challenges

Even though the aforementioned linearity/concavity property makes the policy search a great deal simpler, the exact computation is generally intractable except for relatively small problems. The two major difficulties for exact computation arise from the exponential growth of the vectors with the planning horizon and with the number of observations, and the inefficiencies related to identification of such vectors and
subsequently pruning them. Namely, the number of hyperplanes grows double exponentially such that after \( k \) steps the number of hyperplanes is \( O \left( |U|^{|S|^k} \right) \), where \( |U| \) and \( |S| \) denote the cardinality of the control and observation spaces, respectively. Equivalently, the number of hyperplanes per iteration grows as:

\[
|J^{(k+1)}| = O \left( |U||J^{(k)}||S|^k \right).
\]

This has led to a number of approximations and suboptimal solutions techniques trading off solution quality for speed.

**Remark III.1.** The intractability of the optimal solution for our problem is primarily due to the following reasons:

1) The cost function is minimized over the simplex of probability distributions, i.e., the \((m - 1)\)-dimensional belief simplex for \( m \)-state discrete state-space models.

2) The exponential explosion of the action space with the number of sensors (\( 2^n \) actions).

3) The exponential growth of the \( \alpha \)-vectors with the planning horizon and with the number of observations which itself grows exponentially with the number of active sensors.

4) Due to the data association problem, the number of hypotheses essentially grows exponentially with the number of active sensors for the overlap model in Section II-B.

**C. A point-based sensor scheduler**

To that end, we develop point-based approximate scheduling policies. The key idea is to optimize the value function in (8) only for specific reachable sampled beliefs and not over the entire belief simplex (addressing issue (1) in Remark III.1). Such techniques have shown great potential for solving large scale POMDPs while significantly reducing complexity mostly for robotic applications [10], [15], [17], [19]. There exists a large class of point-based methods which perform point-based backups only at a discrete set of reachable belief points. These algorithms were designed to deal with large state spaces, yet, extra difficulties in the considered scheduling problem arise from the size of the action space \( 2^n \), the observation space, and the number of hypotheses due to data association. To address these difficulties, we devise a strategy to sample actions based on the support of the beliefs and the sparse structure of the transition models. Intuitively speaking, an object can only move from one side of the network to the other side within time constraints rendering exponentially many scheduling actions irrational at certain times. On one hand, this leads to a significant reduction in the action space since instead of performing full updates including \( 2^n \) actions, we perform the minimization over a reduced control space \( U(p) \) for every \( p \in \mathcal{P} \) (see Section III-C1). On the other hand, it also addresses the explosion in the number of hypotheses and the dimensionality of the observation space since, in turn, it caps the number of sensors returning positive observations. At the core of the point-based scheduler we use Perseus [19], a randomized variant of PBVI [15], whereby value iteration updates are not carried out for every sampled belief. Instead, it exploits the key observation that the values for many belief points may be improved simultaneously in one update. Fig. 2 depicts the structure of our point-based approximation, combining control space reduction and data association non-linear filtering with point-based updates. Furthermore, since Perseus updates are not carried out for every sampled belief and multiple belief points are improved simultaneously, the number of \( \alpha \) vectors grows modestly with the number of iterations. This addresses issue (3) in Remark III.1.

First, we briefly outline the steps of Perseus and refer the reader to [16], [19] for further details. Then, we discuss the aforementioned modifications to the algorithm to address the difficulties associated with the dimensionality of the control space and observation space and the number of hypotheses due to data association.

**One iteration of Perseus**

1) Sample a set of belief points \( \mathcal{P} \). We obtain these beliefs by simulating the target motion through the field taking random actions and generating observation according to the observation models in (1), (2), and (3).

2) Sample a belief point \( p \in \mathcal{P} \) at random and compute the backup using (19a) and (19b),

\[
\alpha = \arg \min_{\{\alpha_u\}_{u \in U}} \ p \cdot \alpha_u^p \tag{19a}
\]

where

\[
\alpha_u^p = g(p, u) + \sum_s p(s|u, p) \min \phi(p, u, s) \cdot \alpha_s^{(k)} \tag{19b}
\]

where \( g(p, u) \) is the expected value of the per-stage cost in (5).

3) If \( \sum_b p(b)\alpha(b) \leq J^{(k)}(p) \) then add new \( \alpha \) to \( J^{(k+1)} \) otherwise keep old hyperplane.
4) If \( \{ p \in \mathcal{P} : J^{(k+1)}(p) > J^{(k)}(p) \} = \emptyset \), i.e., the empty set, iteration is complete otherwise repeat from step 1. Fig. 3 illustrates the progress of one iteration of Perseus. The x-axis represents the belief space with circles representing the sampled belief set \( \mathcal{P} = \{ p_1, \ldots, p_r \} \). The y-axis is the value function at consecutive iterations, i.e., \( J^{(k)} \) (solid lines) and \( J^{(k+1)} \) (dashed lines). The figure displays the \( \alpha \) vectors and different steps illustrating the progress of the algorithm. The algorithm selects a belief point at random and updates the value function for that belief. Then a new update is carried out for a belief point randomly selected from the set of remaining beliefs, i.e., beliefs which did not improve in the previous step. The algorithm repeats till all belief points are updated. Solid lines represent the hyper-planes in the \( k \)-th iteration and dashed lines represent the newly added hyper-planes during the \( (k+1) \)-th iteration. In a way, the Perseus updates in POMDPs are the counterpart of asynchronous dynamic programming for MDPs [4] since the order of backup of the belief points is arbitrary and does not require full sweeps over the entire sampled belief set.

1) Gating sensors based on the support of the belief:
Note that the update equation (19) involves a minimization over all control actions in \( \{ \mathcal{U} \} \). Even though one iteration of the algorithm has a complexity that is linear in the cardinality \( |\mathcal{U}| \) of the control space, \( |\mathcal{U}| \) itself is exponential in the number of sensors rendering the minimization infeasible for a relatively large sensor network.

The idea here is to exploit the structure of the scheduling/tracking problem. Since the target transition model is naturally sparse, we predict relatively small uncertainty regions for the target state at future time steps. More specifically, for every belief point in \( \mathcal{P} \), we use prior information about the target transition model to project the future state of the target. This is particularly useful when the current belief vector is sparse leading to more restricted uncertainty regions. Subsequently, we restrict our attention to a significant subset of sensors, that is, sensors of relevance to the particulars of the uncertainty region. Hence, we only consider scheduling actions involving different combinations of a reduced number of sensors, which considerably reduces the control space for every belief in \( \mathcal{P} \). If the number of significant sensors is still large, we randomly sample actions from the reduced control space. Note that the same intuition extends to more complex motion models wherein information about target speed, maneuver, and acceleration can be factored in to define the future uncertainty regions. Hence, instead of performing full updates including \( 2^n \) actions, we perform the minimization over a reduced control space for every \( p \in \mathcal{P} \). Specifically, we redefine the point update equation as:

\[
\alpha = \arg \min_{\{ \alpha^p_u \}_{u \in \mathcal{U}(p)}} p \cdot \alpha^p_u \tag{20}
\]

where \( \mathcal{U}(p) \) designates the reduced control space for the belief vector \( p \).

Note that future iterations of the algorithm involving a particular belief point ensure sufficient sampling to relevant control actions in the reduced control space. This approach is well suited to Perseus wherein the value for every belief point is guaranteed to improve over consecutive stages of the algorithm. It is worth mentioning that the observation and the cost models need to be computed on the fly for each sampled control action during the algorithm implementation. As pointed out earlier, gating a subset of sensors also reduces the number of sensors returning positive observations which in turn reduces the complexity arising from the explosion of the number of observations and number of hypotheses.

IV. Simulation Results

In this section, we show experimental results illustrating the performance of the proposed scheduling policies for the different models considered in this paper. In each simulation run, the object was initially placed at the center of the network and the simulation run concluded when the object reached the absorbing state \( \tau \). We perform Monte Carlo runs to compute the average tracking and energy costs for different values of the energy parameter \( c \). For the planning phase, beliefs are sampled by simulating multiple object trajectories through the sensor network. Each trajectory starts from a random state sampled from the initial belief, picking actions at random, until the target leaves the network.

First, to illustrate some of the basic ideas, consider a simple linear network with 11 sensors where the object moves according to a symmetric random walk either one step to the left or one step to the right. We term this network Net A. Figure 4(a) shows the tradeoff curves between the number of active sensors per unit time and the tracking error per unit time using our point-based scheduler for different levels of clutter density for Net A. As expected, for the no clutter case in the low tracking error regime, i.e., at vanishing energy cost per sensor, activating one sensor to the left or to the right of the sensor is enough to perfectly track the target. The reason being that, at each time step the target would be either perfectly observed (by an awake sensor) or its position can be exactly inferred. Hence, the point-based scheduler in this case converges to the optimal scheduling policy. As the clutter density increases, it is clear that the tracking error increases for the same number of active sensors. The figure illustrates the performance of our point-based scheduler for different clutter densities, namely, when 10% and 20% of the time an active sensor measures clutter. Fig. 4(b) shows the number of active sensors versus the clutter density for fixed energy costs per sensor \( c = 0.1 \) and \( c = 0.2 \). Indeed, for the no clutter case, only one sensor is activated. This is optimal as long as \( c < 0.5 \) since the energy cost is smaller than the expected tracking cost of 0.5. As we increase the clutter density, the scheduler chooses to activate more sensors to compensate for the uncertainty associated with the origin of the measurements until a certain point where the clutter density is just too high that the tracking performance approaches that of uninformed tracking based solely on the available knowledge about the propagation model and the clutter model. In this case, activating sensors is of no avail and the scheduler judiciously chooses to disengage all the
Fig. 3: One iteration of Perseus illustrating the progress of the algorithm. The x-axis represents the belief space with circles representing the sampled belief set $\mathcal{P} = \{p_1, \ldots, p_7\}$. The y-axis is the value function at consecutive iterations, i.e., $J^{(k)}$ and $J^{(k+1)}$. Solid lines represent the hyper-planes at the $k$-th iteration and dashed lines represent the newly added hyper-planes during the $(k+1)$-th iteration. (a) The initial value function $J^{(k)}$; (b) $p_1$ is randomly selected and a new $\alpha$-vector is added to $J^{(k+1)}$. This update step only happens to improve $p_1$. Dark circles represent belief points which did not yet improve; (c) $p_3$ is sampled and a new hyperplane is added which improves the value for $p_2$ through $p_6$; (d) Only $p_7$ did not improve, thus $p_7$ is sampled and a new hyperplane is added to $J^{(k+1)}$; (e) All belief points improved, $J^{(k+1)}$ is computed, the iteration ends.

Fig. 4: (a) Energy tracking tradeoff for different clutter densities for Net A; (b) Number of active sensors versus clutter density for different energy costs per sensor for Net A.
sensors to save the energy resources and avoid unnecessary resource expenditure. Not surprisingly, the x-axis intercept i.e., the cutoff clutter density at which all sensors are deactivated, is larger for smaller values of $c$ since turning on more sensors is less costly.

Second, consider the scenario where the object is moving according to a random walk anywhere from three steps to the left to three steps to the right in each time step. We term this network Net B. The distribution of these movements is given in table I. The change in position indicates movement by a corresponding number of steps to the right or to the left. Fig.5 illustrates the energy-tracking tradeoff of the proposed policies for different levels of clutter density. It is clear that the degradation in performance w.r.t. to the no clutter case is graceful at low and moderate clutter densities.

Next we consider a network where the sensing regions of different sensors overlap. The network consists of 20 possible object locations monitored by 12 sensors as shown in Fig.1. The total cost per unit time versus the energy cost $c$ is shown in Fig.6 (a) for different levels of clutter density. All the curves saturate when the energy cost is too high and the scheduler disengages all the sensors. In this case, the total cost is due to tracking cost based solely on the prior information. Not surprisingly, the saturation point occurs at smaller $c$ for higher values of the clutter density. For moderate values of the clutter density (e.g. 5%), the gap between the saturation points of the cluttered case and the no clutter case is small showing that through judicious use of scheduling actions we are able to compensate for the uncertainty due to clutter. Fig.6 (b) shows the tradeoff curves for this overlap network.

V. CONCLUSIONS

In this paper we developed approximate scheduling strategies for tracking a moving object in cluttered environments using sensor networks. Our approach was to combine Probabilistic Data Association methods for belief update with Point-Based Value Iteration for designing scheduling policies. The generated scheduling policies, albeit suboptimal, provide good energy-tracking tradeoffs.

In the current paper we considered two simple discrete sensing models. Avenues for future research include developing strategies for more realistic models with continuous observation and state spaces, as well as distributed strategies with no central control.

ACKNOWLEDGMENT

This work was funded in part by an Air Force Office of Scientific Research (AFOSR) grant AF FA9550-10-1-0458 and a U.S. Army Research Office MURI grant W911NF-06-1-0094.

REFERENCES

TABLE I: Object movement for Network B.

<table>
<thead>
<tr>
<th>Change in Position</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability</td>
<td>0.3125</td>
<td>0.2344</td>
<td>0.0938</td>
<td>0.0156</td>
</tr>
</tbody>
</table>

![Graph](image1)

Fig. 6: (a) Total cost per unit time versus energy cost per sensor for the overlap network; (b) Energy tracking tradeoff for different clutter densities for network with overlapping sensing ranges.
