Energy Efficient Multi-Object Tracking in Sensor Networks

Jason A. Fuemmeler, *Member, IEEE*, and Venugopal V. Veeravalli, *Fellow, IEEE*

***Abstract*—The problem of tracking multiple objects moving through a network of wireless sensors is studied. It is assumed that each sensor has a limited range for detecting the presence of the object, and that the network is sufficiently dense so that the sensors cover the area of interest. In order to conserve energy the sensors may be put into a sleep mode with a timer that determines the sleep duration. It is assumed that a sensor that is asleep cannot be communicated with or woken up, and hence the sleep duration needs to be determined at the time the sensor goes to sleep based on all the information available to the sensor. The objective is to track the location of the objects to within the accuracy of the range of the sensor. Having sleeping sensors in the network could result in tracking errors, and therefore there is a tradeoff between the energy savings and the tracking errors that result from the sleeping actions at the sensors. Sleeping policies that optimize this tradeoff are designed, and their performance analyzed. This work is an extension of previous work that considered the tracking of only a single object.**

***Index Terms*—Dynamic programming, multi-target tracking, partially observed Markov decision process (POMDP), sensor networks, sleep control.**

I. Introduction

ADVANCES in technology are enabling the deployment of vast sensor networks through the mass production of cheap wireless sensor units with small batteries. Such sensor networks can be used in a variety of application areas. Our focus in this paper is on applications of sensor networks that involve *tracking*, e.g., surveillance, wildlife studies, environmental control, and health care.

We study the problem of tracking multiple objects that are moving through a network of wireless sensors. Each sensor has a limited range for detecting the presence of the objects being tracked, and the objective is to track the location of the objects to within the accuracy of the range of a sensor. For such a tracking problem to be well-posed we need to assume that the sensor field is sufficiently dense so that the sensors cover the entire area of interest. The objects follow random paths through the sensor field whose statistics are assumed to be either known *a priori* or estimated online.

The sensor nodes typically need to operate on limited energy budgets. In order to conserve energy, the sensors may be put into a sleep mode. The use of sleeping sensors in sensor networks for tracking has been studied in the past. It appears that there have been two primary approaches. The first has been to assume that sleeping sensors can be woken up by external means on an as-needed basis (see, e.g., [1]–[6]). Either the method used for this wakeup is left unspecified or it is assumed that there is some low-power wakeup radio at each sensor dedicated to this function. The second approach has involved modifications to power-save functions in MAC protocols for wireless ad hoc networks (see, e.g., [7]–[9]).

In this paper, we wish to examine the fundamental theory of sleeping in sensor networks for tracking, as opposed to the design of protocols for this sleeping. We will assume that the wakeup channel approach is impractical given current sensor technology. In other words, we assume it is not feasible to design a receiver that requires negligible power for operation. Thus, we must consider alternatives to the wakeup channel approach. A straightforward approach is to have each sensor enter and exit the sleep mode using a fixed or a random duty cycle. A more intelligent, albeit more complicated, approach is to use information about the objects' trajectories that is available to the sensor from the network to determine the sleeping strategy. In particular, it is easy to see that the location of the objects (if known) at the time when the sensor is put to sleep would be useful in determining the sleep duration of the sensor; the closer an object is to a sensor, the shorter the sleep duration should be. We take this latter approach in this paper in designing sleeping strategies for the sensors.

In [10], we used the above approach for the tracking of a single object. An optimization problem was formulated that took the form of a partially observable Markov decision process (POMDP). While optimal solutions to this problem could not be found, suboptimal solutions were devised that could be demonstrated to be near optimal.

In this paper, we extend our analysis to the tracking of multiple objects. A discussion of the tracking of multiple objects, often termed multitarget tracking (MTT), can be found in [11]. Tracking multiple objects is not a simple extension of tracking a single object due to the *data association* problem. This problem arises whenever the identity of the objects cannot be determined from the observations. Thus, even if all locations where objects are located are known exactly, it may not be known which location corresponds to which object. This uncertainty leads to

Manuscript received August 24, 2009; accepted March 05, 2010. Date of publication March 29, 2010; date of current version June 16, 2010. This work was supported in part by the U.S. Army Research Office MURI grant W911NF-06-1-0094 through a subcontract from Brown University at the University of Illinois, in part by a grant from the Motorola corporation, and in part by a National Science Foundation Graduate Research Fellowship. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Aleksander Dogandzic.

J. A. Fuemmeler is with the Rockwell Collins, Inc. Cedar Rapids, IA 52402 USA (e-mail: femler1@gmail.com).

V. V. Veeravalli is with the Coordinated Science Lab, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: vvv@illinois.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2010.2046896

an explosion in the set of possibilities that must be considered and makes optimal solution difficult. Suboptimal tracking algorithms are then needed. Examples of such algorithms can be found in [11].

A work of particular interest is the one in [12]. The authors consider energy efficient tracking of multiple objects by selecting duty cycles for the sensors based on the object locations. The work is also interesting in that the statistics of the object movement are not known a priori. The unknown object movement means that they must learn to predict the objects' movements to achieve perfect tracking. Their method of solution is to apply Q-learning to a Markov decision process (MDP). The authors strive for perfect prediction and do not incorporate the case where the object may become lost in their analysis. Such an analysis would require a POMDP formulation, such as the one presented in this paper.

In this paper, our focus is on the design of sleeping policies for use in tracking multiple objects rather than on the tracking performance itself. We therefore formulate a design problem wherein we keep track of the full joint distribution for the object locations. This approach is optimal but can quickly become intractable for large numbers of objects. Our simulation results will focus on the two-object case to make computation as simple as possible. However, most of our analysis applies to the general q-object case and we indicate how our solutions scale with increasing number of objects.

The results of our work are a set of suboptimal sleeping policies. These policies are compared with lower bounds on optimal performance that are derived in the course of our analysis. Our simulation results show how our suboptimal policies compare with optimal performance. Our policies also perform significantly better than naive approaches that do not use information about the locations of the objects. Furthermore, one of our policies uses only information about the marginal distributions of the objects and thus scales well with increasing numbers of objects and can be used in concert with suboptimal tracking algorithms

The remainder of this paper is organized as follows. In Section II, we describe the tracking problem in mathematical terms and define the optimization problem. In Section III, we derive our suboptimal solutions and the associated lower bounds. In Section IV, we provide some numerical results that illustrate the efficacy of the proposed sleeping policies. We summarize and conclude in Section V.

II. Problem Formulation

A. POMDP Formulation

We consider a sensor network with n sensors. For simplicity, we assume that the sensing ranges of the sensors completely cover the region of interest with no overlap. In other words, the region can be divided into n cells with each cell corresponding to the sensing range of a particular sensor. Each sensor can be in one of two states: awake or asleep. A sensor in the awake state consumes more energy than one in the asleep state. However, object sensing can be performed only in the awake state. An awake sensor can only detect whether one or more objects is within its range and can detect neither the exact number of objects present nor which objects are present. A more complicated model for sensing would include the possibility of multiple sensors receiving simultaneous noisy observations of the objects. However, this would complicate the analysis considerably and so we stick with our simpler model.

The movement of each object to be tracked is described by a first-order Markov chain whose state is the current location of the object to within the accuracy of a cell. However, we also append an additional state that occurs when the object leaves the network. Thus, there are $n+1$ possible states for each object with states 1 through n occurring while the object is in the network and state $n+1$ representing the appended state. We impose the constraint that the object must leave the network with probability 1. Once state $n+1$ is reached, the object remains in that state. The statistics for each object movement can be described using a $(n+1)\times(n+1)$ probability transition matrix. We make the simplifying assumption that state $n+1$ can always be observed, regardless of the sensor states. To this end, we define a "sentry" sensor at location $n+1$ that is always awake but consumes no energy. This sentry sensor is like other sensors in that it cannot determine which of the objects have left the network. Our models for the movement of the objects are simplistic, but do allow us to investigate sleeping policy design. The principles we develop later in this paper should extend to more complicated object movement models.

We are interested in tracking q objects that move independently according to their individual first-order Markov models. We will write the combined state of the q objects as a vector of length q. There are $(n+1)^q$ possible states for this vector. The state $\boldsymbol{\mathcal{T}} \equiv [n+1,\ldots,n+1]$ is the terminal state that occurs when all objects have left the network. Once this state is reached, no further cost is incurred. We define a kernel P such that $P(\boldsymbol{x},\boldsymbol{y})$ is the probability that the next state is $\boldsymbol{y}$ given that the current state is $\boldsymbol{x}$. We can predict t time steps into the future by defining $P^1 = P$ and P^t inductively as

$$P^t(\boldsymbol{x},\boldsymbol{y}) = \sum_{z\in\{1,\ldots,n+1\}^q} P^{t-1}(\boldsymbol{x},\boldsymbol{z})P(\boldsymbol{z},\boldsymbol{y}). \tag{1}$$

Suppose p is a function on $\{1,\ldots,n+1\}^q$ such that $p(\boldsymbol{x})$ is the probability that the state is $\boldsymbol{x}$ at the current time step. Then the probability that the state will be $\boldsymbol{y}$ at t time steps in the future is given by

$$(pP^t)(\boldsymbol{y}) \equiv \sum_{x\in\{1,\ldots,n+1\}^q} p(\boldsymbol{x})P^t(\boldsymbol{x},\boldsymbol{y}). \tag{2}$$

We also use the notation $\delta_{\boldsymbol{b}}$ to denote a function such that

$$\delta_{\boldsymbol{b}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} = \boldsymbol{b} \\ 0, & \text{else.} \end{cases} \tag{3}$$

Let $\boldsymbol{b}_k$ denote the state for the objects at time k. Conditioned on $\boldsymbol{b}_k$, the distribution for $\boldsymbol{b}_{k+1}$ is given as

$$\boldsymbol{b}_{k+1} \sim \delta_{\boldsymbol{b}_k} P \tag{4}$$

i.e., $\boldsymbol{b}_{k+1}$ is distributed according to the probability mass function $\delta_{\boldsymbol{b}_k}P$. This defines the evolution of the object locations.

To provide a means for centralized control, we assume the presence of an extra node called the central controller. The central controller keeps track of the state of the network and assigns

sleep times to sensors that are awake. In particular, each sensor that wakes up remains awake for one time unit during which the following actions are taken: 1) the sensor sends a binary observation to the central unit that indicates whether one or more objects are within its range, and 2) the sensor receives a new sleep time (which may equal zero) from the central controller. The sleep time input is used to initialize a timer at the sensor that is decremented by one time unit each time step. When this timer expires, the sensor wakes up. Since we assume that wakeup signals are impractical, this timer expiration is the only mechanism for waking a sensor.

Let $r_{k,\ell}$ denote the value of the sleep timer of sensor ℓ at time k. We call the $(n+1)$-vector $\boldsymbol{r}_k$ the residual sleep times of the sensors at time k. Also, let $u_{k,\ell}$ denote the sleep time input supplied to sensor ℓ at time k. We add the constraints $r_{k,n+1}=0$ and $u_{k,n+1}=0$ due to the nature of the sentry sensor at location $n+1$. We can describe the evolution of the residual sleep times as

$$r_{k+1,\ell} = (r_{k,\ell}-1)\mathbb{1}_{\{r_{k,\ell}>0\}} + u_{k,\ell}\mathbb{1}_{\{r_{k,\ell}=0\}} \tag{5}$$

for all k and $\ell \in \{1,\ldots,n+1\}$. The first term on the right-hand side of this equation expresses that if the sensor is currently asleep (the sleep timer for the sensor is not zero), the sleep timer is decremented by 1. The second term expresses that if the sensor is currently awake (the sleep timer is zero), the sleep timer is reset to the current sleep time input for that sensor.

Based on (4) and (5), we see that we have a discrete-time dynamical model that describes our system. The *state* of the system at time k is described by $x_k = (\boldsymbol{b}_k, \boldsymbol{r}_k)$ and the state evolution is defined in (4) and (5). Unfortunately, not all of x_k is known to the central unit at time k since $\boldsymbol{b}_k$ is known only if the object locations are being tracked precisely. Thus, we have a dynamical system with incomplete (or partially observed) state information. If we denote the observation available to the central unit at time k by z_k, then we have

$$z_k = (\boldsymbol{s}_k, \boldsymbol{r}_k) \tag{6}$$

where $\boldsymbol{s}_k$ is a $(n+1)$-vector of observations with

$$s_{k,\ell}=\begin{cases} 0, & \text{if } r_k,\ell{=}0 \text{ and no objects are at locations } \ell \\ 1, & \text{if } r_k,\ell{=}0 \text{ and one or more objects are at locations } \ell \\ \mathcal{E}, & \text{if } r_k,\ell{>}0 \end{cases} \tag{7}$$

where $\mathcal{E}$ is an erasure that provides no information. The total information available to the control unit at time k is given by

$$I_k = (z_0,\ldots,z_k,\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{k-1}) \tag{8}$$

with $I_0 = z_0$ denoting the initial (known) state of the system. The control input for sensor ℓ at time k is allowed to be a function of I_k, i.e.,

$$\boldsymbol{u}_k = \mu_k(I_k). \tag{9}$$

The vector-valued function μ_k is the sleeping policy at time k.

We now identify the costs present in our tracking problem. The first is an *energy cost* of $c>0$ for each sensor that is awake. This model is used for simplicity, although it would perhaps be better to take vary the cost at each sensor as remaining battery life decreases. The energy cost can be written mathematically as

$$\sum_{\ell=1}^{n} c\mathbb{1}_{\{r_{k,\ell}=0\}}. \tag{10}$$

The second cost is *tracking error*. We can further decompose tracking error into two components. The first component is *observation error* that occurs when we fail to observe a particular object. The second component is *data association error* that occurs when the objects have been misidentified. To perform the object identification, we define the vector of estimated object locations at time k to be $\hat{\boldsymbol{b}}_k$. We can think of $\hat{\boldsymbol{b}}_k$ as an additional control input that is a function of I_k, i.e.,

$$\hat{\boldsymbol{b}}_k = \beta_k(I_k). \tag{11}$$

Since $\hat{\boldsymbol{b}}_k$ does not affect the state evolution, we do not need to include past values of this control input in I_k. We combine observation and data association errors by defining a tracking error to have occurred when either an observation error or a data association error has occurred. A cost of 1 is incurred for each tracking error. Thus the tracking cost can be written as

$$\sum_{i=1}^{q}\left(1-\mathbb{1}_{\{r_{k,b_{k,i}}=0\}}\mathbb{1}_{\{\hat{b}_{k,i}=b_{k,i}\}}\right). \tag{12}$$

Note that we need not have included observation error (note that an observation error occurs for the ith object whenever $r_{k,b_{k,i}} \neq 0$) in our tracking cost since an object can sometimes be located through a process of elimination without being observed. However, we include observation error to make the problem easier to separate later on. The parameter c is used to trade off energy consumption and tracking errors.

Recall that the $\hat{\boldsymbol{b}}_k$ input does not affect the state evolution; it only affects the cost. We can therefore compute the optimal choice of $\hat{\boldsymbol{b}}_k$, denoted as $\beta_k^*(I_k)$, using an optimization minimizing the tracking error over a single time step. We can thus write

$$\beta_k^*(I_k) = \arg\min_{\hat{\boldsymbol{b}}} \mathsf{E}\left[\sum_{i=1}^{q}\left(1-\mathbb{1}_{\{r_{k,b_{k,i}}=0\}}\ \mathbb{1}_{\{\hat{b}_i=b_{k,i}\}}\right)\middle| I_k\right]. \tag{13}$$

Remembering that once the terminal state is reached no further cost is incurred, we can write the total cost for time step k as

$$g(\boldsymbol{b}_k, I_k) = \mathbb{1}_{\{\boldsymbol{b}_k\neq\boldsymbol{T}\}}\left(\sum_{\ell=1}^{n} c\mathbb{1}_{\{r_{k,\ell}=0\}} + \sum_{i=1}^{q}(1-\mathbb{1}_{\{r_{k,b_{k,i}}=0\}}\mathbb{1}_{\{\beta_k^*(I_k)=b_{k,i}\}})\right). \tag{14}$$

The infinite horizon cost for the system is given by

$$J(I_0,\mu_0,\mu_1,\ldots) = \mathsf{E}\left[\sum_{k=1}^{\infty} g(\boldsymbol{b}_k, I_k)\middle| I_0\right]. \tag{15}$$

Since g is bounded by $(cn+q)$ and the expected time till the object leaves the network is finite, the cost function J is well-defined. The goal is to compute the solution to

$$J^*(I_0) = \min_{\mu_0,\mu_1,\ldots} J(I_0,\mu_0,\mu_1,\ldots). \tag{16}$$

The solution to this optimization problem for each value of c yields an optimal sleeping policy. The optimization problem falls under the framework of a POMDP.

B. Dealing With Partial Observability

Partial observability presents a problem since the information for decision-making at time k given in (8) is unbounded in memory. To remedy this, we seek a sufficient statistic for optimization that is bounded in memory. We see from (7) that the vector of observations $\boldsymbol{s}_k$ depends only on the state x_k, which in turn depends only on x_{k-1}, u_{k-1}, and w_{k-1}. It is a standard argument (e.g., see [13]) that for such an observation model, a sufficient statistic is given by the probability distribution of the state x_k given I_k. Such a sufficient statistic is referred to as a *belief state* in the POMDP literature (e.g., see [14] and [15]). Since the residual sleep times portion of our state is observable, the sufficient statistic can be written as $v_k = (p_k, \boldsymbol{r}_k)$, where p_k is a probability mass function over $\{1,\ldots,n+1\}^q$. Mathematically, we have

$$p_k(\boldsymbol{b}) = \mathsf{P}(\boldsymbol{b}_k = \boldsymbol{b} | I_k). \tag{17}$$

The evolution of p_k is difficult to write mathematically, but it is a standard nonlinear filtering operation. The computation at time $k+1$ can be described procedurally as follows.

1) Form $p' = p_k P$. This is the distribution before incorporating the observations.
2) For all $\boldsymbol{b}$ such that the z_{k+1} is inconsistent with a state of $(\boldsymbol{b}, \boldsymbol{r}_{k+1})$, change the value of $p'(\boldsymbol{b})$ to 0.
3) Normalize p' to create a probability distribution. This new distribution is p_{k+1}.

One example of a distribution update is shown in Fig. 1. For notational convenience, we also define

$$p_{k,i}(b) = \sum_{\boldsymbol{b}':b'_i=b} p_k(\boldsymbol{b}'). \tag{18}$$

In other words, $p_{k,i}$ is the marginal distribution for object i.

The function β_k^* that determines $\hat{\boldsymbol{b}}_k$ can now be written in terms of p_k and $\boldsymbol{r}_k$ instead of I_k. We can rewrite it as

$$\beta_k^*(p_k,\boldsymbol{r}_k) = \arg\min_{\hat{\boldsymbol{b}}} \mathsf{E}\left[\sum_{i=1}^{q}\left(1 - \mathbb{1}_{\{r_{k,b_{k,i}}=0\}} \times \mathbb{1}_{\{\hat{b}_i=b_{k,i}\}}\right) | \boldsymbol{b}_k \sim p_k\right] \tag{19}$$

$$= \arg\min_{\hat{\boldsymbol{b}}} \sum_{\boldsymbol{b}}\left(p_k(\boldsymbol{b}) \times \sum_{i=1}^{q}(1 - \mathbb{1}_{\{r_{k,b_i}=0\}}\mathbb{1}_{\{\hat{b}_i=b_i\}})\right) \tag{20}$$

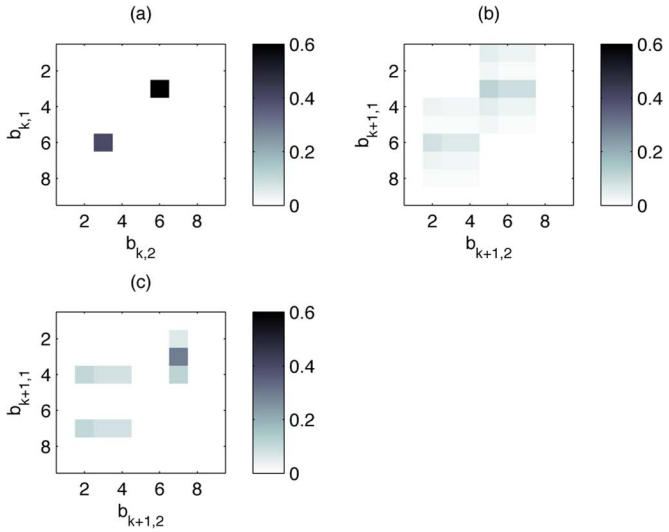

Fig. 1. An example of a distribution update for $q = 2$ and $n = 9$. In each subfigure, a joint distribution for the objects is shown. In (a), it is known that one object is located at position 3 and one is located at position 6. In (b), the joint distribution at time $k+1$ before incorporating observations is shown. In generating (c), we suppose that sensors 1, 5, 6, and 8 are awake and have failed to observe the object. The distribution in (c) is the one that results from incorporating these observations.

$$= \arg\max_{\hat{\boldsymbol{b}}} \sum_{i=1}^{q}\sum_{\boldsymbol{b}} p_k(\boldsymbol{b})\mathbb{1}_{\{r_{k,b_i}=0\}}\mathbb{1}_{\{\hat{b}_i=b_i\}}. \tag{21}$$

Thus, each component of the vector valued function β_k^* can be chosen according to

$$\beta_{k,i}^*(p_k,\boldsymbol{r}_k) = \arg\max_{\hat{b}} \sum_{\boldsymbol{b}} p_k(\boldsymbol{b})\mathbb{1}_{\{r_{k,b_i}=0\}}\mathbb{1}_{\{\hat{b}=b_i\}} \tag{22}$$

$$= \arg\max_{\hat{b}} \mathbb{1}_{\{r_{k,\hat{b}}=0\}} p_{k,i}(\hat{b}). \tag{23}$$

In other words, for each object we select the estimated object location from among the locations where a sensor is awake. From these locations, we select the one with the largest value of the marginal distribution for that object. Note that β_k^* has the same form for every k so we can drop the subscript and refer to β_k^* as β^*.

We now wish to write our dynamic programming problem in terms of the sufficient statistic. We first wish to rewrite the cost at time step k. Note that since only expected values of the cost function g appear in (15), we can take our cost function to be the expected value of g [defined in (14)] conditioned on $\boldsymbol{b}_k$ being distributed according to p_k. Abusing notation, we call this redefined cost g. The cost can be written as

$$g(p_k,\boldsymbol{r}_k) = \sum_{\boldsymbol{b}} p_k(\boldsymbol{b})\mathbb{1}_{\{\boldsymbol{b}\neq\boldsymbol{\mathcal{T}}\}}\left(\sum_{\ell=1}^{n} c\mathbb{1}_{\{r_{k,\ell}=0\}} + \sum_{i=1}^{q}(1 - \mathbb{1}_{\{r_{k,b_i}=0\}}\mathbb{1}_{\{\beta^*(p_k,\boldsymbol{r}_k)=b_i\}})\right) \tag{24}$$

or more simply as

$$g(p_k,\boldsymbol{r}_k) = \sum_{\boldsymbol{b}\neq\boldsymbol{\mathcal{T}}} p_k(\boldsymbol{b})\left(\sum_{\ell=1}^{n} c\mathbb{1}_{\{r_{k,\ell}=0\}} + \sum_{i=1}^{q}(1 - \mathbb{1}_{\{r_{k,b_i}=0\}}\mathbb{1}_{\{\beta^*(p_k,\boldsymbol{r}_k)=b_i\}})\right). \tag{25}$$

However, from (23) we have that $\beta^*(p_k, \boldsymbol{r}_k) = b_i$ implies $r_{k,b_i} = 0$. We can therefore rewrite this cost as

$$g(p_k, \boldsymbol{r}_k) = \sum_{\boldsymbol{b} \neq \boldsymbol{\mathcal{T}}} p_k(\boldsymbol{b}) \left(\sum_{\ell=1}^{n} c \mathbb{1}_{\{r_{k,\ell}=0\}} + \sum_{i=1}^{q} \left(1 - \mathbb{1}_{\{\beta^*(p_k, \boldsymbol{r}_k)=b_i\}}\right) \right). \quad (26)$$

The selection of sleep times, originally presented in (9), can now be rewritten as

$$\boldsymbol{u}_k = \mu_k(p_k, \boldsymbol{r}_k). \quad (27)$$

The total cost defined in (15) becomes

$$J(p_0, \boldsymbol{r}_0, \mu_0, \mu_1, \ldots) = \mathsf{E}\left[\sum_{k=1}^{\infty} g(p_k, \boldsymbol{r}_k) | v_0 \right] \quad (28)$$

and the optimal cost defined in (16) becomes

$$J^*(p_0, \boldsymbol{r}_0) = \min_{\mu_0, \mu_1, \ldots} J(p_0, \boldsymbol{r}_0, \mu_0, \mu_1, \ldots). \quad (29)$$

III. Suboptimal Solutions

Similar to the single object case in [10], an optimal policy could be found by solving the Bellman equation

$$J(p, \boldsymbol{r}) = \min_{\mu} \mathsf{E}\Big[g(p_1, \boldsymbol{r}_1) + J(p_1, \boldsymbol{r}_1) | p_0 = p, \boldsymbol{r}_0 = \boldsymbol{r}, \boldsymbol{u}_0 = \mu(p_0, \boldsymbol{r}_0) \Big]. \quad (30)$$

However, since an optimal solution could not be found in [10] for the simpler single object case, we immediately turn our attention to finding suboptimal solutions to our problem.

Our approach to generating suboptimal solutions is similar to that used in [10]. Namely, we make unrealistic assumptions that greatly simplify the evolution of p_k. These assumptions also allow the tracking and energy costs to be written as a sum of costs, one for each sensor. The result is that the problem then separates into a set of n simpler subproblems, one for each sensor, that can be more easily solved. The solution to the ℓth subproblem defines the policy for sensor ℓ. The performance of the resultant policy depends on the assumptions made in the simplification. In this paper, we do not include the mathematical details of deriving the n simpler subproblems, but the procedure is straightforward and closely follows that in [10].

A. Q_{MDP} Policy

In the POMDP literature (e.g., see [14] and [15]), a Q_{MDP} solution is one in which it is assumed that the partially observed state becomes fully known after a control input has been chosen. Note that this assumption implies that there will be no future data association errors and thus the only tracking costs present in designing this policy are observation errors. As a result, the per-sensor Bellman equation for sensor ℓ is given by

$$J^{(\ell)}(p) = \min_{u} \sum_{\boldsymbol{b} \neq \boldsymbol{\mathcal{T}}} \left(\sum_{j=1}^{u} (pP^j)(\boldsymbol{b}) \sum_{i=1}^{q} \mathbb{1}_{\{b_i=\ell\}} + c(pP^{u+1})(\boldsymbol{b}) + (pP^{u+1})(\boldsymbol{b}) J^{(\ell)}(\delta_{\boldsymbol{b}}) \right). \quad (31)$$

Note that the three terms inside the minimization represent the tracking cost, the energy cost, and the future cost, respectively, given a sleep time of u. Thus, (31) is a Bellman equation for the per-sensor problem under the assumption of perfect future observations.

This equation can be solved through the use of policy iteration. Initially, a value is assigned to the function $J^{(\ell)}$ for each possible state. The right-hand side of (31) is applied to compute a new value for $J^{(\ell)}$ for each possible state. This process is continued until convergence is obtained. Note that the number of states for our problem is $(n+1)^q$. Thus, this solution may not scale well as the number of sensors or the number of objects becomes large.

Note that for the Q_{MDP} solution, we are assuming more information than is actually available. Thus, the cost function obtained under the Q_{MDP} assumption is a lower bound on optimal performance. We will use this lower bound when we present our numerical results.

B. First Cost Reduction (FCR) Policy

Suppose we assume that no observations will be available in the future so that the evolution of p_k is a straightforward application of the kernel P. Unfortunately, we find that the problem does not separate under this assumption. This is because data association errors do not allow the tracking cost to be written as a sum of tracking costs for each sensor. To achieve separation, we can decide to use a tracking cost that only incorporates observation errors. If we do this, then the per-sensor Bellman equation for sensor ℓ is given by

$$J^{(\ell)}(p) = \min_{u} \sum_{\boldsymbol{b} \neq \boldsymbol{\mathcal{T}}} \left(\sum_{j=1}^{u} (pP^j)(\boldsymbol{b}) \sum_{i=1}^{q} \mathbb{1}_{\{b_i=\ell\}} + c(pP^{u+1})(\boldsymbol{b}) + J^{(\ell)}(pP^{u+1}) \right). \quad (32)$$

Note that the three terms inside the minimization represent the tracking cost, the energy cost, and the future cost, respectively, given a sleep time of u. Thus, (32) is a Bellman equation for the per-sensor problem under the assumption of no future observations.

It is easy to verify that

$$J^{*(\ell)}(p) = \sum_{j=1}^{\infty} \min \left\{ \sum_{\boldsymbol{b} \neq \boldsymbol{\mathcal{T}}} (pP^j)(\boldsymbol{b}) \sum_{i=1}^{q} \mathbb{1}_{\{b_i=\ell\}}, \sum_{\boldsymbol{b} \neq \boldsymbol{\mathcal{T}}} c(pP^j)(\boldsymbol{b}) \right\} \quad (33)$$

is indeed a solution to (32). In other words, at each time step we incur a cost that is the minimum of the expected observation cost at sensor ℓ and the expected energy cost at sensor ℓ. The sleeping policy for sensor ℓ is to select the first value of u such that

$$\sum_{\boldsymbol{b}\neq\boldsymbol{\mathcal{T}}}(pP^{u+1})(\boldsymbol{b})\sum_{i=1}^{q}\mathbb{1}_{\{b_i=\ell\}}\geq\sum_{\boldsymbol{b}\neq\boldsymbol{\mathcal{T}}}c(pP^{u+1})(\boldsymbol{b}). \tag{34}$$

This gives rise to the name first cost reduction (FCR), since a sensor comes awake only when the expected tracking cost when not awake first exceeds the expected energy cost when awake. Because policy iteration over a large number of states is not required, the FCR policy does not have the scaling problems associated with the Q_{MDP} policy.

However, by slightly altering the policy we can make it easier to implement. Note that in the right-hand side of (34), a cost of c is incurred if at least one object remains in the network. We can approximate the right-hand side of (34) by instead assuming we incur a cost of c/q for *each* object still in the network. This approximation can be written as

$$\sum_{\boldsymbol{b}\neq\boldsymbol{\mathcal{T}}}c(pP^{u+1})(\boldsymbol{b})\approx\sum_{i=1}^{q}\frac{c}{q}\sum_{\boldsymbol{b}:b_i\neq n+1}(pP^{u+1})(\boldsymbol{b}). \tag{35}$$

Also note that the left-hand side of (34) can be rewritten as

$$\sum_{\boldsymbol{b}\neq\boldsymbol{\mathcal{T}}}(pP^{u+1})(\boldsymbol{b})\sum_{i=1}^{q}\mathbb{1}_{\{b_i=\ell\}}=\sum_{i=1}^{q}\sum_{\boldsymbol{b}:b_i=\ell}(pP^{u+1})(\boldsymbol{b}). \tag{36}$$

Thus, we can define a new policy (which we term the FCR policy) that is to select the first value of u such that

$$\sum_{i=1}^{q}\sum_{\boldsymbol{b}:b_i=\ell}(pP^{u+1})(\boldsymbol{b})\geq\sum_{i=1}^{q}\sum_{\boldsymbol{b}:b_i\neq n+1}\frac{c}{q}(pP^{u+1})(\boldsymbol{b}). \tag{37}$$

Note that the inner summation on both sides of this inequality can be written in terms of the marginal distributions for object i associated with pP^{u+1}. Because the objects move independently, the marginal distributions in the absence of observations also evolve independently. Note that we ignore the joint statistics for the objects only when generating the sleeping policy; for tracking we must keep track of the joint statistics to incorporate the information from the observations. In suboptimal MTT algorithms that only approximate the joint distribution (see [11] for examples), it is usually still possible to determine the marginals for each object and our FCR policy can be used in concert with these algorithms.

C. All Awake (AA) Policy

The lower bound that results from the Q_{MDP} policy is likely to be loose when data association errors dominate the tracking cost. In this section, we design a Q_{MDP}-like policy that, instead of assuming the state is known, assumes the following:

- at the current time step, after selecting sleep times all sleeping sensors will be allowed to make observations (with no energy cost);
- at future time steps, the distribution for the object location will evolve as if all sensors are awake.

This gives rise to the term All Awake (AA policy). Note that the AA assumption is like assuming we will have "perfect observations." However, this does not imply perfect knowledge of the state due to the presence of data association errors. Note that since we are assuming more information than is actually available, the AA assumption does yield a lower bound on optimal performance. Due to complexity issues, we will only design the AA policy for the $q=2$ case.

The advantage of the AA assumption is it allows us to considerably simplify the state space for p_k. Since all the sensors come awake at each time step, the set of at most two locations where an object could be present is known exactly. Suppose for the moment that there are two distinct locations where an object is observed. Let $\tilde{\boldsymbol{b}_k}=(\tilde{b}_{k,1},\tilde{b}_{k,2})$ with $\tilde{b}_{k,1}<\tilde{b}_{k,2}$ being the locations where objects are present at time k. Thus, $\tilde{\boldsymbol{b}_k}$ belongs to a subset of $\{1,\ldots,n+1\}^2$. To completely characterize p_k we need only specify the probability that $\boldsymbol{b}_k=\tilde{\boldsymbol{b}}_k$. Denote this probability as d_k. Then with probability $1-d_k$ we have that $\boldsymbol{b}_k=(\tilde{b}_{k,2},\tilde{b}_{k,1})$. Note that if there is only one distinct location where an object is observed we can simply let $\tilde{b}_{k,1}=\tilde{b}_{k,2}$ and $d_k=1$.

Let $\tilde{x}_k=(\tilde{\boldsymbol{b}}_k,d_k)$. The state space for $\tilde{x}_k$ is not finite due to $d_k\in[0,1]$. The approach we take is to quantize d_k and construct a kernel $\tilde{P}$ for this quantized version of $\tilde{x}_k$. Note that in doing this we no longer have a true lower bound; however, with fine enough quantization we can well approximate such a lower bound. Note that at the time when sleep times are selected, the distribution p_k may not be able to be written in the form $(\tilde{\boldsymbol{b}}_k,d_k)$. However, we can compute the probability that a particular value of $\tilde{x}_k$ will be the result of allowing all sensors to wake up in the current time step. We can thus form a distribution $\tilde{p}_k$ from p_k to be used in the dynamic programming.

Even with these simplifications, the problem still does not separate into a subproblem for each sensor. To see this, suppose that $\tilde{b}_{k,1}<\tilde{b}_{k,2}$. Then the expected tracking cost at time k is given by

$$\begin{aligned}&2\mathbb{1}_{\{r_{k,\tilde{b}_{k,1}}>0\}}\mathbb{1}_{\{r_{k,\tilde{b}_{k,2}}>0\}}\\&\quad+1(\mathbb{1}_{\{r_{k,\tilde{b}_{k,1}}=0\}}\mathbb{1}_{\{r_{k,\tilde{b}_{k,2}}>0\}}+\mathbb{1}_{\{r_{k,\tilde{b}_{k,1}}>0\}}\mathbb{1}_{\{r_{k,\tilde{b}_{k,2}}=0\}})\\&\quad+2\min\{d_k,1-d_k\}\mathbb{1}_{\{r_{k,\tilde{b}_{k,1}}=0\}}\mathbb{1}_{\{r_{k,\tilde{b}_{k,2}}=0\}}.\end{aligned} \tag{38}$$

This cannot be written in a linear form that allows us to separate the actions of the sensor at location $\tilde{b}_{k,1}$ from those at $\tilde{b}_{k,2}$. We would like to have separation so that we can eliminate the residual sleep times and solve a set of simpler problems. To achieve this and still yield a lower bound, we lower bound the expected tracking cost expression as

$$\begin{aligned}&2\min\{d_k,1-d_k\}+(1-2\min\{d_k,1-d_k\})\\&\qquad\times\left(\mathbb{1}_{\{r_{k,\tilde{b}_{k,1}}>0\}}+\mathbb{1}_{\{r_{k,\tilde{b}_{k,2}}>0\}}\right).\end{aligned} \tag{39}$$

Note that this lower bound is tight as long as at least one of the sensors is awake. Note also that if $\tilde{b}_{k,1}=\tilde{b}_{k,2}$ (which implies that $d_k=1$) then we have that the expected tracking cost can be written as

$$2\mathbb{1}_{\{r_{k,\tilde{b}_{k,1}}>0\}} \tag{40}$$

which it turns out can also be written in the form of (39).

To better see how this analysis leads to separation, we first define functions T^W and T^S such that $T^W(\tilde{x}_k, \ell)$ and $T^S(\tilde{x}_k, \ell)$ are the tracking costs incurred by sensor ℓ when it is awake and asleep, respectively. We define these functions as

$$T^W(\tilde{x}_k, \ell) = \frac{1}{n} 2\min\{d_k, 1 - d_k\} \tag{41}$$

and

$$T^S(\tilde{x}_k, \ell) = \frac{1}{n} 2\min\{d_k, 1 - d_k\} + (1 - 2\min\{d_k, 1 - d_k\}) \left(\mathbb{1}_{\{\tilde{b}_{k,1} = \ell\}} + \mathbb{1}_{\{\tilde{b}_{k,2} = \ell\}} \right). \tag{42}$$

It is a simple matter to verify that

$$\sum_{\ell=1}^{n} \left(T^W(\tilde{x}_k, \ell) \mathbb{1}_{\{r_{k,\ell} = 0\}} + T^S(\tilde{x}_k, \ell) \mathbb{1}_{\{r_{k,\ell} > 0\}} \right) \tag{43}$$

reduces to (39). Furthermore, this equation clearly separates into a tracking cost associated with each sensor.

The per-sensor Bellman equation for sensor ℓ under the above assumptions is given by

$$J^{(\ell)}(\tilde{p}) = \min_{u} \sum_{\tilde{x} \neq \tilde{T}} \left(\sum_{j=1}^{u} (\tilde{p}\tilde{P}^j)(\tilde{x}) T^S(\tilde{x}, \ell) + (\tilde{p}\tilde{P}^{u+1})(\tilde{x}) T^W(\tilde{x}, \ell) + c(\tilde{p}\tilde{P}^{u+1})(\tilde{x}) + (\tilde{p}\tilde{P}^{u+1})(\tilde{x}) J^{(\ell)}(\delta_{\tilde{x}}) \right). \tag{44}$$

This equation can be solved through the use of policy iteration to yield a policy and a lower bound. Due to the larger size and irregular structure of the $\tilde{P}$ kernel, optimizing the right-hand side of (44) over all $u \geq 0$ is made difficult. In our simulation results, we will optimize over $0 \leq u \leq u_{\max}$ where $u_{\max}$ is sufficiently large. This results in further approximation to the actual AA lower bound.

IV. Numerical Results

In this section, we give some simulation results that illustrate the performance of the policies we derived in previous sections. These results will be for one-dimensional sensor networks, but the general behavior should extend to two-dimensional networks. In each simulation run, the objects were initially placed at the center of the network and the locations of the objects made known to each sensor. This initial condition would not occur in practice, but was used for simplicity. Alternatively, we could assume a partially unknown initial state, but the trends described in the simulation results below are expected to remain unchanged. A simulation run concluded when all objects left the network. The results of many simulation runs were then averaged to compute an average tracking cost and an average energy cost. To allow for easier interpretation of our results, we then normalized our costs by dividing by the *expected* time for a simulation run. We refer to these normalized costs as costs per unit time, even though the true costs per unit time would use the *actual* times for each simulation run (the difference between the two was found to be small).

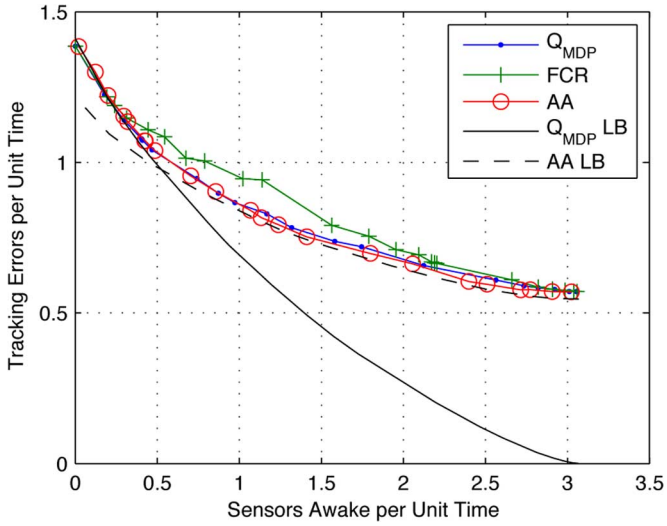

Fig. 2. Tradeoff curves for Q_{MDP}, FCR, and AA policies for Network A and $a = 0.55$.

We first consider a network we term Network A. This network is a one-dimensional network with seven sensors. The small number of sensors was needed because the AA policy must perform policy iteration for a number of states equal to $(n(n+1)/2)\lambda + n + 1$ where λ is the number quantization levels for d_k. In our simulations, we used $\lambda = 21$ for a total of 596 states. Policy iteration for this number of states required significant computation and the network could not be made much larger. A value of $u_{\max} = 50$ was used in computing the AA policy. The object movement in Network A is parametrized by a scalar $a \in [0, 1]$. Object 1 moves one cell to the left with probability a and one cell to the right with probability $1 - a$ in each time step. Object 2 does just the opposite, moving one cell to the left with probability $1 - a$ and one cell to the right with probability a. Note that the closer a is to 0.5, the more difficult it is to distinguish between the objects based on their movements. This means that by varying a we can investigate the performance of our policies for various amounts of data association error.

We illustrate the performance of our policies for Network A for the cases $a = 0.55$, $a = 0.75$, and $a = 0.95$ in Figs. 2–4, respectively. Curves are shown for the Q_{MDP} and AA lower bounds as well as the Q_{MDP}, FCR, and AA policies. The curves are tradeoff curves that examine the tradeoff between energy cost and tracking cost as the parameter c is varied. In examining the tradeoff curves, the distance from the right-hand point of each curve to the x-axis is the average number of data association errors when all sensors are awake. No tracking error smaller than this can be achieved. From the figures we can draw the following conclusions.

- The lower bound due to the Q_{MDP} assumption is tight when only a few sensors are awake (large c). This is because the Q_{MDP} assumption incorporates only observation errors and when few sensors are making observations, observation errors dominate.
- The lower bound due to the AA assumption is tight only when many sensors are awake (small c). This is because

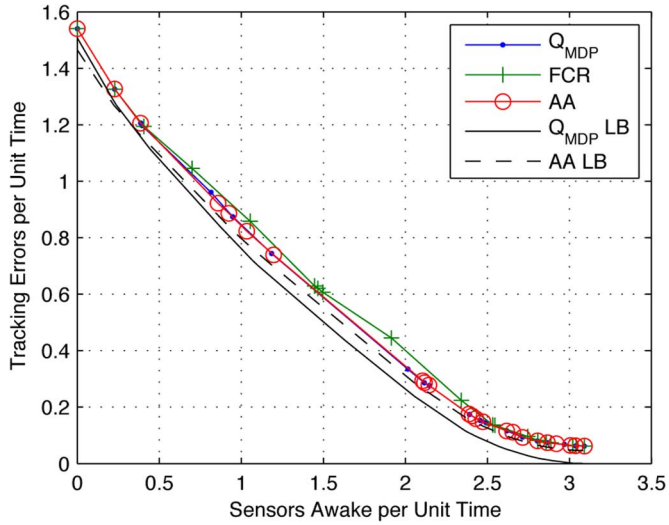

Fig. 3. Tradeoff curves for Q_{MDP}, FCR, and AA policies for Network A and $a = 0.75$.

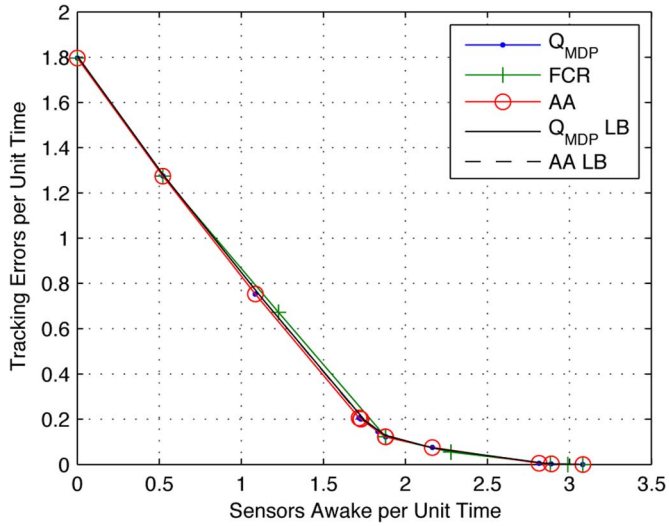

Fig. 4. Tradeoff curves for Q_{MDP}, FCR, and AA policies for Network A and $a = 0.95$.

the tracking cost approximation used in computing the AA policy is loose when neither object is observed. Also, remember that the AA lower bound plotted is actually an approximation to the true lower bound due to quantization effects and the use of a finite $u_{\max}$.

- The Q_{MDP} policy performs best when only a few sensors are awake, and the AA policy performs best when many sensors are awake. Not surprisingly, these are the same regions where their bounds are tight.
- The FCR policy is the worst-performing policy. The difference between the FCR policy and the other policies, while never especially large in terms of the tradeoff curves, shrinks as data association errors become small.

We now turn our attention to simulating the FCR policy, which scales better than the other policies, for a larger network, termed Network B. Network B is a one-dimensional network with 41 sensors. Note that policy iteration for the Q_{MDP} policy would need to be performed over $(n+1)^2 = 1764$ states and the requirements for the AA policy would be even larger.

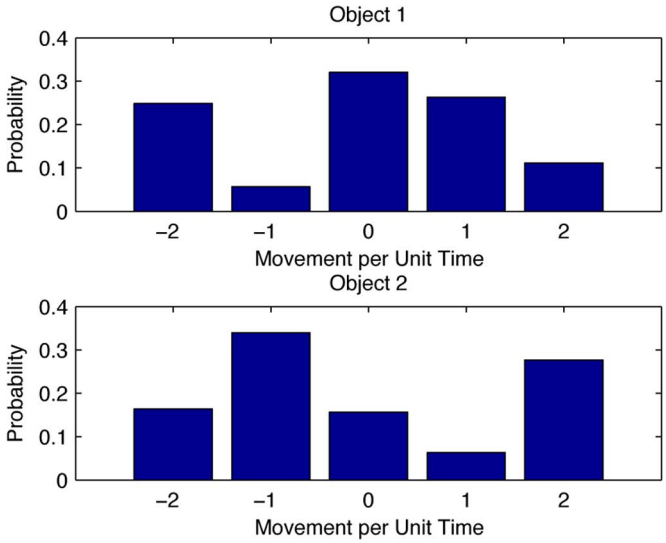

Fig. 5. Object movement distributions for Network B.

TABLE I
OBJECT MOVEMENT DISTRIBUTIONS FOR NETWORK B

Change in Position	-2	-1	0	+1	+2
Probability for Object 1	0.2482	0.0568	0.3205	0.2633	0.1112
Probability for Object 2	0.1641	0.3395	0.1566	0.0633	0.2765

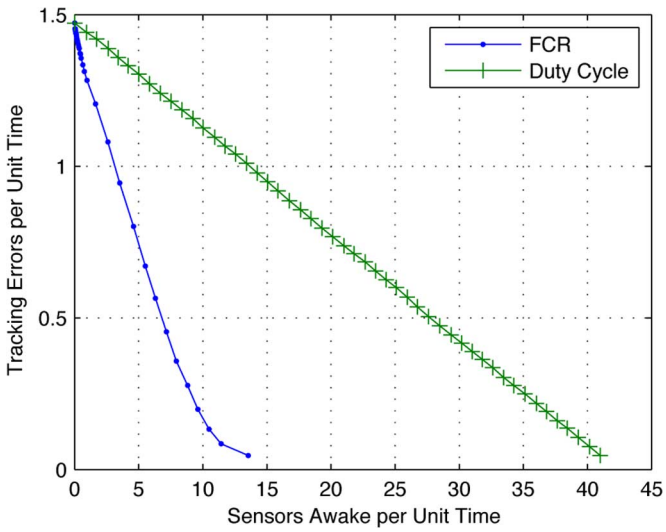

Fig. 6. Tradeoff curves for FCR and duty cycle policies for Network B.

The distributions for the movement of the objects are given in Table I and illustrated graphically in Fig. 5.

Since no lower bounds are available for Network B, we compare the performance of our FCR policy to a duty cycle policy, where each sensor comes awake with some fixed probability at each time step. Fig. 6 shows tradeoff curves for these two policies. The tradeoff curve for the duty cycle policy is generated by varying the probability that a sensor is awake between 0 and 1. The FCR tradeoff curve appears reasonable and significantly outperforms the duty cycle policy.

V. CONCLUSION

In this paper, we formulated a problem for tracking multiple objects in an energy-efficient manner. We found that while an

optimal solution could not be found, it was possible to design suboptimal solutions that approximate optimal performance, as seen in our simulation results. Our results also indicate that the tradeoff between energy consumption and tracking errors can be considerably improved by using information about the location of the object. Of the policies we designed, the FCR policy exhibits the best scaling laws although there is some loss in performance.

There are several avenues for future research. Since the models for sensing, object movement, and energy usage used in this paper were simplistic, more sophisticated models need to be examined. Distributed strategies for the scenario where a central controller is not available is another area for future research. Finally, solving the tracking problem when the statistics for object movement are unknown or partially known presents another interesting challenge.

REFERENCES

[1] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proc. IEEE*, vol. 91, no. 8, pp. 1163–1171, Aug. 2004.

[2] S. Balasubramanian, I. Elangovan, S. K. Jayaweera, and K. R. Namuduri, "Distributed and collaborative tracking for energy-constrained ad-hoc wireless sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2004, vol. 3, pp. 1732–1737.

[3] R. Gupta and S. R. Das, "Tracking moving targets in a smart sensor network," in *Proc. IEEE 58th Veh. Technol. Conf.*, Oct. 2003, vol. 5, pp. 3035–3039.

[4] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proc. 1st IEEE Int. Workshop Sens. Netw. Protocols Applicat.*, May 2003, pp. 71–81.

[5] Y. Xu, J. Winter, and W.-C. Lee, "Prediction-based strategies for energy saving in object tracking sensor networks," in *Proc. IEEE Int. Conf. Mobile Data Manage.*, 2004, pp. 346–357.

[6] L. Yang, C. Feng, J. W. Rozenblit, and J. Peng, "A multi-modality framework for energy efficient tracking in large scale wireless sensor networks," in *Proc. IEEE Int. Conf. Netw., Sens., Control*, Apr. 2006, pp. 916–921.

[7] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proc. 10th Annu. Int. Conf. Mobile Comput. Netw. (MOBICOM)*, Sep. 2004, pp. 129–143.

[8] C. Gui and P. Mohapatra, "Virtual patrol: A new power conservation design for surveillance using sensor networks," in *Proc. 4th Int. Symp. Inf. Process. Sens. Netw. (IPSN)*, Apr. 2005, pp. 246–253.

[9] N. A. Vasanthi and S. Annadurai, "Energy saving schedule for target tracking sensor networks to maximize the network lifetime," in *Proc. 1st Int. Conf. Commun. Syst. Software and Middleware*, Jan. 2006, pp. 1–8.

[10] J. A. Fuemmeler and V. V. Veeravalli, "Smart sleeping policies for energy efficient tracking in sensor networks," *IEEE Trans. Signal Process.* , vol. 56, no. 5, pp. 2091–2101, May 2008.

[11] J. Liu, M. Chu, and J. E. Reich, "Multitarget tracking in distributed sensor networks," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 36–46, May 2007.

[12] W.-L. Yeow, C.-K. Tham, and W.-C. Wong, "Energy efficient multiple target tracking in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, pp. 918–928, Mar. 2007.

[13] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA: Athena Scientific, 2007.

[14] D. Aberdeen, "A (revised) survey of approximate methods for solving partially observable Markov decision processes," National ICT Australia, Canberra, Australia, 2003 [Online]. Available: http://users.rsise.anu.edu.au/~daa/papers.html

[15] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 362–370.

Jason Fuemmeler (M'07) received the B.E.E. degree in electrical engineering from the University of Dayton, Dayton, OH, in 2000 and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign in 2004 and 2008, respectively.

He has been awarded a NSF graduate research fellowship and a Vodafone fellowship. He is currently employed in the Advanced Technology Center at Rockwell Collins performing research in electronic warfare and wireless communications.

Venugopal V. Veeravalli (M'92–SM'98–F'06) received the B.Tech. degree (Silver Medal Honors) from the Indian Institute of Technology, Bombay, in 1985, the M.S. degree from Carnegie Mellon University, Pittsburgh, PA, in 1987, and the Ph.D. degree from the University of Illinois at Urbana-Champaign, in 1992, all in electrical engineering.

He joined the University of Illinois at Urbana-Champaign in 2000, where he is currently a Professor in the Department of Electrical and Computer Engineering and a Research Professor in the Coordinated Science Laboratory. He served as a Program Director for communications research at the U.S. National Science Foundation, Arlington, VA, from 2003 to 2005. He has previously held academic positions at Harvard University, Rice University, and Cornell University, and has been on sabbatical at MIT, IISc Bangalore, and Qualcomm, Inc. His research interests include distributed sensor systems and networks, wireless communications, detection and estimation theory, and information theory.

Prof. Veeravalli is a Distinguished Lecturer for the IEEE Signal Processing Society for 2010–2011. He has been on the Board of Governors of the IEEE Information Theory Society. He has been an Associate Editor for Detection and Estimation for the IEEE TRANSACTIONS ON INFORMATION THEORY and for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. Among the awards he has received for research and teaching are the IEEE Browder J. Thompson Best Paper Award, the National Science Foundation CAREER Award, and the Presidential Early Career Award for Scientists and Engineers (PECASE).