EFFICIENT TRACKING IN A NETWORK OF SLEEPY SENSORS

Venugopal V. Veeravalli and Jason A. Fuemmeler

University of Illinois at Urbana-Champaign
ECE Department and Coordinated Science Laboratory
1308 West Main Street
Urbana, Illinois, 61801-2307

ABSTRACT

We study the problem of tracking an object that is moving randomly through a dense network of wireless sensors. We assume 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. We assume that a sensor that is asleep cannot be communicated with or woken up. Thus 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 object to within the accuracy of the range of the sensor. However, having sleeping sensors in the network could result in tracking errors, and hence there is a tradeoff between the energy savings and the tracking error that results from the sleeping actions at the sensors. We consider the design of sleeping policies that optimize this tradeoff.

Keywords: wireless sensors, sleep mode, random walk, dynamic programming.

1. 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 an object that is moving through a network of wireless sensors as shown in Figure 1. Each sensor has a limited range for detecting the presence of the object being tracked, and the objective is to track the location of the object to within the accuracy of the range of the 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 object follows a random path through the sensor field whose statistics are assumed to be either known *a priori* or estimated on-line.

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. One way to effect the transition between the active and sleep modes is to have the sensor be in sleep mode by default and to use a low-power radio channel to wake up (page) the sensor when it is needed. To the best of our knowledge, this approach has been taken in most of the literature to date on energy-efficient tracking using sensor networks (see, e.g., [1–7]). The design problem that we consider is very different in that we assume that using a wake-up channel is *impractical* given current sensor technology.

This research was supported by the National Science Foundation under the award CCF-0049089, through the University of Illinois, and by the Motorola Center for Communications at the University of Illinois.

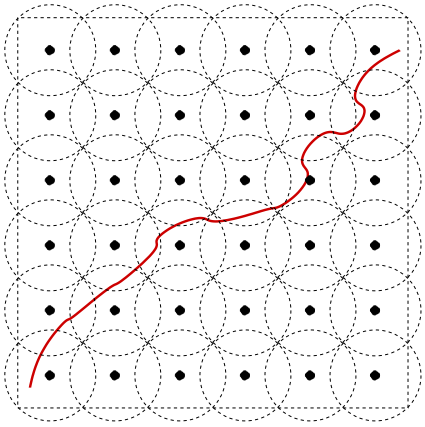

Fig. 1. Object tracking in a field of sensors.

There are alternatives to the wake-up channel approach. For example, we may 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 object trajectory 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 object (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 the object, the shorter the sleep duration should be. We take this latter approach in this paper in designing sleeping strategies for the sensors.

It is clear that having sleeping sensors in the network could result in tracking errors, and hence there is a tradeoff between the energy savings and the tracking error that results from the sleeping at the sensors. The sleeping policies at the sensors should be designed to optimize this tradeoff. In order to simplify the optimization problem, we assume that there is a central unit that controls the sensor network. All information about the object trajectory is stored at this central unit and is used to determine sleep times of sensors that come awake.

The remainder of this paper is organized as follows. In Section 2, we describe the tracking problem in mathematical terms and define the optimization problem. In Section 3, we outline a dynamic programming approach to finding the optimal solution. However, we find that the size of state space renders the optimization intractable for networks of more than a handful of sensors. We therefore pro-

pose some practical approximations in Section 4. In the course of deriving these approximations, we also develop a lower bound on optimal performance that allows us to characterize sleeping policy performance. In Section 5, we provide some numerical results that illustrate the efficacy of the proposed sleeping policies. We summarize and conclude in Section 6.

2. PROBLEM DESCRIPTION

The essential features of the tracking problem described in Figure 1 are contained in a one-dimensional simplification where the sensors are placed on a line and the object undergoes a random walk on the line. We hence consider this simplification in the remainder of the paper to facilitate presentation, with the understanding that the techniques that we develop can be generalized to the two-dimensional tracking problem.

Consider a one-dimensional sensor network with sensors placed unit distance apart from $-m$ to $+m$. An object that has to be tracked by this sensor network is assumed to undergo a random walk along the line. Let b_k denote the location of the object at time k. Then

$$b_{k+1} = b_k + w_k \tag{1}$$

where $\{w_k\}$ are i.i.d. integer-valued random variables with known distribution. We assume that $w_k \in [-n, n]$ with n typically being much smaller than m. For example, $\{w_k\}$ could be i.i.d. Bernoulli random variables that take the value $+1$ or -1 with equal probability. The tracking problem stops when the object leaves the network, i.e., when $b_k \notin \{-m, \ldots, 0, \ldots, m\}$.

A central unit, which controls this sensor network, is assumed to maintain the information required to compute the sleep times of the sensors in the system and to assign the sleep times for the sensors that come awake. A sensor is either awake or asleep at each time instant. Each sensor that wakes up remains awake for one time unit during which the following actions are taken: (i) if the object is within its range, the sensor detects the object and sends this information to the central unit, and (ii) the sensor receives a new sleep time (which may equal zero) from the central controller. The input from the central unit is used to set a sleep timer at the sensor, which gets decremented by one every time unit.

Let $r_{k,\ell}$ denote the residual sleep time at time k for the sensor located at position ℓ, i.e., $r_{k,\ell}$ is the value of the sleep timer at sensor ℓ at time k. Also let $u_{k,\ell}$ denote the control input (sleep time) given to sensor ℓ from the central unit at time k. We can write the update of $r_{k,\ell}$ as

$$r_{k+1,\ell} = (r_{k,\ell} - 1)\mathbb{1}_{\{r_{k,\ell}>0\}} + u_{k,\ell}\mathbb{1}_{\{r_{k,\ell}=0\}} \tag{2}$$

where $\mathbb{1}$ is the indicator function. We use the vector notation $\boldsymbol{r}_k = (r_{k,-m}, \ldots, r_{k,m})$ and $\boldsymbol{u}_k = (u_{k,-m}, \ldots, u_{k,m})$.

Based on (1) and (2), we see that we have discrete-time dynamical model that describes our tracking problem, with exogenous input w_k and control input $\boldsymbol{u}_k$. The *state* of the system at time k is described by $x_k = (b_k, \boldsymbol{r}_k)$ and it has the following evolution in time:

$$x_{k+1} = \begin{cases} f(x_k, \boldsymbol{u}_k, w_k) & \text{if } x_k \neq \mathcal{T} \\ \mathcal{T} & \text{if } x_k = \mathcal{T} \text{ or if } b_k \notin \{-m, \ldots, m\} \end{cases} \tag{3}$$

where $\mathcal{T}$ denotes a terminal state that the system reaches when the objects exits the sensor network, and f is described by (1) and (2). Once in the terminal state, the system remains there indefinitely. With some possible abuse of notation, we denote the components of the terminal state corresponding to both b_k and $\boldsymbol{r}_k$ by $\mathcal{T}$.

Unfortunately, not all of x_k is known to the central unit at time k since b_k is known only if the sensor at location b_k is awake at time k. Thus we have 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 $z_k = (s_k, \boldsymbol{r}_k)$, with

$$s_k = \begin{cases} b_k & \text{if } b_k \neq \mathcal{T} \text{ and } r_{k,b_k} = 0 \\ \mathcal{E} & \text{if } b_k \neq \mathcal{T} \text{ and } r_{k,b_k} > 0 \\ \mathcal{T} & \text{if } b_k = \mathcal{T} \end{cases} \tag{4}$$

where $\mathcal{E}$ denotes an unknown or "erasure" value.

The total information available to the control unit at time k is given by

$$I_k = (z_0, \ldots, z_k, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1}). \tag{5}$$

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.,

$$u_{k,\ell} = \mu_{k,\ell}(I_k). \tag{6}$$

We assume that an energy cost of unity is contributed by each sensor that is awake, and a tracking cost of c is incurred for each time unit that the object is not tracked. The total cost at time k is then given by

$$g(x_k) = \mathbb{1}_{\{x_k \neq \mathcal{T}\}} \left[c\,\mathbb{1}_{\{r_{k,b_k}>0\}} + \sum_{\ell=-m}^{m} \mathbb{1}_{\{r_{k,b_k}=0\}} \right]. \tag{7}$$

Thus c is the parameter used to tradeoff energy consumption and tracking errors, and the total cost values for different values of c produce the tradeoff curve for a given sleeping policy.

The total cost (over a possibly infinite horizon trajectory) for the system is given by

$$J_0(I_0, \mu_0, \mu_1, \ldots) = \mathsf{E}\left[\sum_{k=1}^{\infty} g(x_k) \middle| I_0 \right] \tag{8}$$

Since g is bounded by $(2m + 1 + c)$, the cost function J_0 is guaranteed to be bounded as long as the expected time for the object to exit the system is finite. The latter condition holds for any nontrivial random walk. Hence the following optimization problem is well defined.

$$J_0^*(I_0) = \min_{\mu_0, \mu_1, \ldots} J_0(I_0, \mu_0, \mu_1, \ldots) \tag{9}$$

The solution to this optimization problem for each value of c yields an optimal sleeping policy. The optimization problem falls under the framework of partially observable Markov decision process (POMDP), and the optimal solution may be obtained via dynamic programming (DP).

3. OPTIMAL SOLUTION VIA DP

3.1. Sufficient statistic for DP

The information for decision-making at time k given in (5) is unbounded in memory. It is easy to show via standard arguments (see, e.g. [8]) that a sufficient statistic for optimization, that is bounded in memory, is given by the probability distribution of the state x_k, given I_k. Since $\boldsymbol{r}_k$ is part of x_k, the sufficient statistic can be written as $v_k = (\boldsymbol{r}_k, \boldsymbol{p}_k)$, where $\boldsymbol{p}_k$ is a row vector that denotes the probability distribution of the location of the object, b_k, given I_k. The components of $\boldsymbol{p}_k$ are given by:

$$p_{k,\ell} = \mathsf{P}(\{b_k = \ell\}|I_k), \quad \ell = -m, \ldots, m, \tag{10}$$

and $p_{k,m+1} = \mathsf{P}(\{b_k = \mathcal{T}\}|I_k)$.

The sufficient statistic (or belief state as it is referred to in the POMDP literature [9]) can be updated recursively based on the new observation. It is easiest to see this in two steps. First we update $\boldsymbol{p}_k$ without using the new observation z_{k+1}, i.e., using only I_k to form vector $\boldsymbol{q}_{k+1}$ with components

$$q_{k+1,\ell} = \mathsf{P}(\{b_{k+1} = \ell\}|I_k) \tag{11}$$

and $q_{k+1,m+1} = \mathsf{P}(\{b_{k+1} = \mathcal{T}\}|I_k)$. The vector $\boldsymbol{q}_{k+1}$ is obtained from $\boldsymbol{p}_k$ via a Markov evolution with transition matrix $\mathbb{P}$ defined by statistics of the jump variables $\{w_k\}$:

$$\boldsymbol{q}_{k+1} = \boldsymbol{p}_k \, \mathbb{P} \tag{12}$$

The last row of $\mathbb{P}$ corresponds to the absorbing terminal state.

We now "clean up" $\boldsymbol{q}_{k+1}$ using the new observation z_{k+1} as follows. If the object is observed at sensor ℓ we replace $\boldsymbol{q}_{k+1}$ with a unit point mass at ℓ. If the object is not observed by any of the sensors that are awake, we zero out the those components of $\boldsymbol{q}_{k+1}$ and normalize the remaining ones. Thus

$$p_{k+1,\ell} = \mathbb{1}_{\{s_{k+1}=\ell\}} + \mathbb{1}_{\{s_{k+1}=\mathcal{E}\}} \, \mathbb{1}_{\{r_{k+1,\ell}\neq 0\}} \cdot \cdot \frac{q_{k+1,\ell}}{\sum_i \mathbb{1}_{\{r_{k+1,i}\neq 0\}} q_{k+1,i}}. \tag{13}$$

3.2. Tractability of optimal solution

We can easily write down the finite-horizon DP equations in terms of the sufficient statistic $v_k = (\boldsymbol{r}_k, \boldsymbol{p}_k)$. Furthermore, it is easily established that the finite-horizon cost-to-go functions converge as the horizon goes to infinity and that the corresponding limits are independent of k due to the stationary nature of the problem. Thus the optimal cost in (9) is given by the infinite-horizon cost-to-go function, and the corresponding optimal control functions μ_k are the same for all k. The optimal cost and the optimal sleeping policy can hence be found by solving a Bellman equation [8], via known techniques such as successive approximation. However, the optimal solution is intractable even for small sensor networks. This is because the state space grows exponentially with the number of sensors. For example, even with seven sensors with maximum sleep time of only 10 and probability mass function quantized to multiples of 0.1, there are about 10^9 possible states v_k.

4. PRACTICAL APPROXIMATIONS

4.1. $\mathrm{Q}_{\mathrm{MDP}}$ Solution

Because the optimal solution is intractable, we wish to formulate an alternative problem that is tractable yet retains most of the essential features of the optimal soltion. A popular approach to finding good suboptimal solutions for POMDP's is to assume that at times after the current time, we will have perfect state information. The solution so obtained is known in the literature as the $\mathrm{Q}_{\mathrm{MDP}}$ solution [9].

We assume that beyond the current time, each sensor somehow knows the exact position of the object each time it wakes up. Thus, whenever a sensor wakes up, the set of possible distributions it sees is the set of point mass distributions. Under this assumption it is clear that from the perspective of a sensor ℓ, the actions of the other sensors do not affect the state evolution. We also know that a sensor ℓ can only affect the cost that accrues either when sensor ℓ comes awake or when a tracking error occurs at sensor ℓ. Thus, the optimization problem under this assumption fully separates into $2m+1$ problems — one for each sensor.

Let us solve the optimization problem at sensor ℓ using an infinite-horizon dynamic program. Since the residual sleep times of the other sensors are irrelevant to optimal decision making in the $\mathrm{Q}_{\mathrm{MDP}}$ setting, the sufficient statistic for decision making at time k is simply $\boldsymbol{p}_k$. The Bellman equation for this problem is easily shown to be

$$J^{(\ell)}(\boldsymbol{p}) = \min_{\mu^{(\ell)}} \quad \Big(\sum_{i=1}^{u} c \left[\boldsymbol{p}\mathbb{P}^i\right]_\ell + \sum_{j\neq\mathcal{T}} \left[\boldsymbol{p}\mathbb{P}^{u+1}\right]_j + \sum_k \left[\boldsymbol{p}\mathbb{P}^{u+1}\right]_k J^{(\ell)}(\boldsymbol{e}_k) \Big)\Big|_{u=\mu^{(\ell)}(\boldsymbol{p})} \tag{14}$$

where $\boldsymbol{e}_b$ denotes a row vector with a one in position b and zeros everywhere else, and where $J^{(\ell)}$ is the infinite-horizon cost-to-go function for sensor ℓ. The $\mathrm{Q}_{\mathrm{MDP}}$ policy for sensor ℓ, $\mu_Q^{(\ell)}$, is given from the minimization on the RHS of (14).

If we can solve (14) for $\boldsymbol{p} = \boldsymbol{e}_b$ for $b \in \{-m, \ldots, m, \mathcal{T}\}$, we have sufficient information to define the solution for any other distribution $\boldsymbol{p}$. Thus we have $2m+2$ equations in $2m+2$ unknowns. However, this set of equations does not have a unique solution since we can add an arbitrary constant to a solution $J^{(\ell)}$ and still satisfy the equations. We therefore add the additional constraint that $J^{(\ell)}(\boldsymbol{e}_\mathcal{T}) = 0$ (which is clearly the desired solution). This reduces the problem to one of $2m+1$ equations in $2m+1$ unknowns with a unique solution. An effective method for finding the solution is to use policy iteration [9].

4.2. A lower bound on optimal performance

Since the $\mathrm{Q}_{\mathrm{MDP}}$ solution assumes more information than is actually available, the cost obtained in its derivation is a lower bound on the cost of any scheme. In particular, if we apply the $\mathrm{Q}_{\mathrm{MDP}}$ policy to the actual system (without perfect state information), we will achieve a higher cost. Intuitively, the lower bound should be tightest when the number of tracking errors is small so that the assumption that the position of the object is known is most realistic.

4.3. Point mass approximations

The $\mathrm{Q}_{\mathrm{MDP}}$ policy, $\mu_Q = \{\mu_Q^{(\ell)} : \forall \ell\}$ is considerably easier to compute than the optimal policy and can be computed on-line after some initial off-line computation has been completed. However, such on-line computation requires sufficient processing power and could introduce delays. It would be convenient if $\mu_Q^{(\ell)}$ could be pre-computed and stored either at the central controller or at sensor ℓ itself. The latter option is particularly attractive since it allows for decentralized implementation. But the set of possible distributions $\boldsymbol{p}$ is potentially quite large — even if quantization is performed — and could make the storage requirements prohibitive.

To make the storage requirements feasible, we consider approximations of the $\mathrm{Q}_{\mathrm{MDP}}$ algorithm where $\boldsymbol{p}$ is replaced by a unit point mass distribution. There are two options for the placement of the unit point mass: (i) the centroid of $\boldsymbol{p}$, and (ii) the nearest point to the sensor on the support of $\boldsymbol{p}$. The latter option allows for the implementation of the $\mathrm{Q}_{\mathrm{MDP}}$ policy without detailed information about the statistics of the random walk – only the support of the jump variables w_k is required!

5. NUMERICAL RESULTS

Simulations of the various polices were performed for 1-D sensor networks. In these simulations, the object was initially placed at the center of the networtk and the location of the object was made known to each sensor. By averaging over many simulation runs, it was possible to compute the average number of tracking errors and the average number of sensors awake per unit time. These values

could then be plotted for different values of c to generate a tradeoff curve for these two quantities.

Figures 2 and 3 show results for two different networks. The results of Figure 2 are for a network with 41 sensors ($m = 20$) where the object moved according to a symmetric random walk. In other words, the $\{w_k\}$ were i.i.d. random variables taking on value $+1$ or -1 with equal probability. The results of Figure 3 are for a network with 61 sensors ($m = 30$) where the $\{w_k\}$ were i.i.d. random variables uniformly distributed over $\{-3, -2, \ldots, 2, 3\}$.

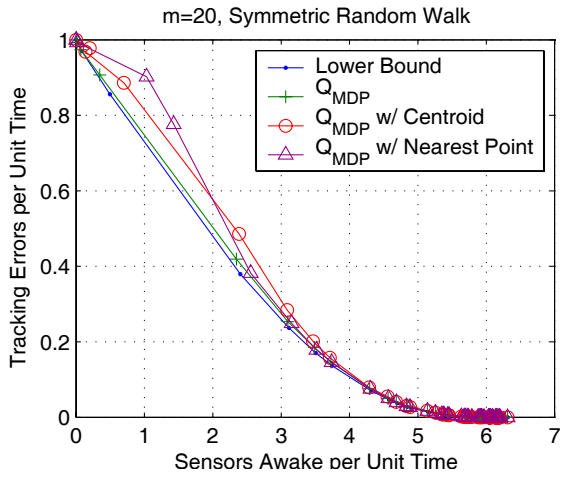

Fig. 2. Comparison of lower bound and Q_{MDP} solutions for $m = 20$.

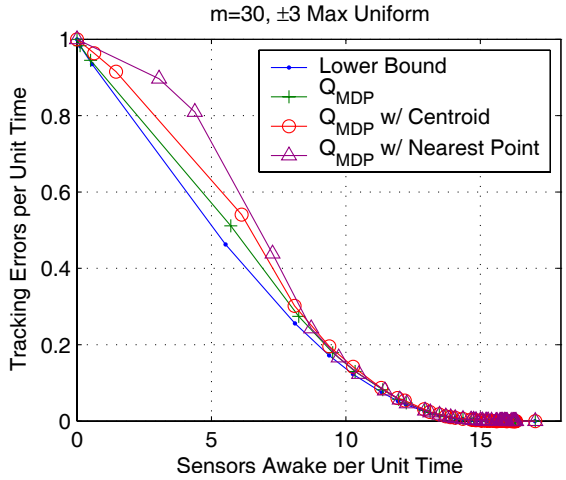

Fig. 3. Comparison of lower bound and Q_{MDP} solutions for $m = 30$.

Four curves are plotted in each figure. The first curve is the tradeoff curve that results from the lower bound described in Section 4.2. Although this curve is unachievable, it is useful as a baseline since if a sleeping policy approaches the performance of this reference curve, that sleeping policy must also be approaching optimal performance. The remaining three curves are simulation results for the Q_{MDP} solution and for the Q_{MDP} solution using the centroid and nearest point approximations described in Section 4.3.

From these simulation results, we see that the Q_{MDP} solution is very close to the curve for our lower bound and is thus nearly optimal. This is especially true in the regime of interest where the tracking error is small. The use of point mass approximations does result in some loss of performance, but again this loss is small for small tracking error.

Note that we could also consider a more primitive policy (which does not use location information) where each sensor would be awake with some probability π at each time instant. As π were varied, we would achieve a tradeoff curve that is a straight line between the points $(0, 1)$ and $(2m + 1, 0)$ in the coordinate system used in the above plots. When compared with this tradeoff curve, the schemes we have proposed result in significant improvement.

We have obtained similar results for a variety of other cases for the object trajectory, including one-dimensional walks with more complicated statistics for w_k, and two-dimensional random walks, which we could not present in this paper due to space limitations.

6. CONCLUSION

We studied a tracking problem with sleepy sensors and showed that the tradeoff between energy consumption and tracking errors can be considerably improved by using information about the location of the object. In particular significant savings in energy are possible by allowing for some tracking errors. The Q_{MDP} approach appears to be very promising in that near optimal performance can be obtained at reasonable complexity. Avenues for further research include decentralized strategies for the scenario where a central controller is not available, and adaptive strategies for the case where the statistics of the object's trajectory are unknown.

7. REFERENCES

[1] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-based Collaboration for target tracking in sensor networks," *IEEE Trans. on Wireless Comm.*, vol. 3, no. 5, pp. 1689–1701, Sep 2004.

[2] W. Zhang and G. Cao, "An energy efficient framework for mobile target tracking in sensor networks," in *IEEE MILCOM*, 2003, vol. 1, pp. 597–602.

[3] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proc. of the IEEE*, vol. 91, no. 8, pp. 1163–1171, August 2003.

[4] S. Balasubramanian et al, "Distributed and collaborative tracking for energy-constrained ad-hoc wireless sensor networks," in *IEEE WCNC*, 2004, vol. 3, pp. 1732–1737.

[5] R. Gupta and S. R. Das, "Tracking moving targets in a smart sensor network," in *IEEE WCNC*, 2004, vol. 3, pp. 3035–3039.

[6] Y. Xu and W-C. Lee, "On localized prediction for power efficient object tracking in sensor networks," in *Proc. of 23rd Int. Conf. on Distributed Computing Systems*, 2003, pp. 434–439.

[7] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proc. IEEE Int. Workshop on Sensor Network Protocols and Applications*, 2003, pp. 71–81.

[8] D. Bertsekas, *Dynamic Programming*, Prentice-Hall, Upper Saddle River, NJ, 1987.

[9] D. Aberdeen, "A (revised) survey of approximate methods for solving POMDP's," *Technical Report*, Dec. 2003, http://users.rsise.anu.edu.au/ daa/papers.html.