Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 31520, Pages 1–18
DOI 10.1155/ASP/2006/31520

Nonmyopic Sensor Scheduling and its Efficient Implementation for Target Tracking Applications

Amit S. Chhetri,[1] Darryl Morrell,[2] and Antonia Papandreou-Suppappola[1]

[1] *Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287, USA*
[2] *Department of Engineering, Arizona State University, Tempe, AZ 85287, USA*

Received 12 May 2005; Revised 1 October 2005; Accepted 8 November 2005

Recommended for Publication by Joe C. Chen

We propose two nonmyopic sensor scheduling algorithms for target tracking applications. We consider a scenario where a bearing-only sensor is constrained to move in a finite number of directions to track a target in a two-dimensional plane. Both algorithms provide the best sensor sequence by minimizing a predicted expected scheduler cost over a finite time-horizon. The first algorithm approximately computes the scheduler costs based on the predicted covariance matrix of the tracker error. The second algorithm uses the unscented transform in conjunction with a particle filter to approximate covariance-based costs or information-theoretic costs. We also propose the use of two branch-and-bound-based optimal pruning algorithms for efficient implementation of the scheduling algorithms. We design the first pruning algorithm by combining branch-and-bound with a breadth-first search and a greedy-search; the second pruning algorithm combines branch-and-bound with a uniform-cost search. Simulation results demonstrate the advantage of nonmyopic scheduling over myopic scheduling and the significant savings in computational and memory resources when using the pruning algorithms.

1. INTRODUCTION

In recent years, advances in sensor technology coupled with embedded systems and wireless networking has made it possible to deploy sensors in numerous applications including environmental science, defense information, and security. A critical component of sensor technology is maximizing the sensing utility while minimizing the cost of sensing resources. Sensor scheduling, a method to allocate future sensing resources by optimizing a performance metric over a finite time-horizon, can be an effective solution to this problem. The performance metric may differ depending on the system application: tracking accuracy in target tracking, battery power or communication bandwidth in a network of low-power sensor motes, or an amount of information gained in surveillance.

The sensor scheduling problem can be formulated as a stochastic control problem that involves optimization of an expected scheduler cost over time. Although dynamic programing can be used to obtain optimal closed-loop solutions [1, 2], computing these solutions is often prohibitively expensive, and suboptimal open-loop or greedy approaches are used instead [3, 4]. Previous work on sensor scheduling for target tracking can be found in [4–7]. In [4], the scheduling is myopic (one step ahead) and maximizes the Rényi information for binary-valued measurements. In [5], the sensors are scheduled by maximizing the mutual information between the state estimate and the measurement sequence. The scheduling is performed over a continuous state space using a stochastic approximation approach in [6], whereas in [7], the scheduling chooses the least number of sensors necessary to reduce the covariance matrix of estimate error to a desired value. Recently, a posterior Cramér-Rao lower bound-(PCRLB) based scheduling method was applied to multisensor resource deployment [8] and sensor trajectory planning [9]. The objective was to deploy fixed multiple sensors and determine sensor trajectories in a bearing-only tracking scenario by optimizing scheduler costs based on the predicted Fisher information matrix.

Sensor scheduling is *nonmyopic* if it is performed multiple steps ahead in the future. As we will demonstrate, although myopic scheduling has lower computational costs than nonmyopic scheduling, in some cases it performs worse than nonmyopic scheduling. For example, in [10], nonmyopic scheduling significantly improved the performance for target tracking in a sensor network.

For sensor scheduling problems in which the configuration at any given time epoch is selected from one of a finite number of options, the use of nonmyopic sensor scheduling can be difficult. This is because the computational time and memory requirements of the optimal scheduler can increase exponentially with the time horizon. The computational burden could be reduced using pruning algorithms. Such algorithms have been studied extensively in artificial intelligence and operations research in [11–13] and in the context of sensor scheduling in [5, 14]. In [5], an information-theoretic-based pruning algorithm was derived for linear Gaussian systems and applied suboptimally to nonlinear Gaussian systems. In [14], sliding-window and threshold methods were proposed to increase search efficiency. Note, however, that these scheduling approaches are not guaranteed to find the best sensor sequence.

In this paper, we consider sensor scheduling problems in which there is a finite set of possible sensor configurations at each time epoch. We make two main contributions to this problem. First, we propose two nonmyopic sensor scheduling algorithms for target tracking applications that can be implemented with different scheduler costs. Second, we propose the use of two branch-and-bound- (B&B) based pruning algorithms to significantly reduce the computational burden of the scheduling algorithms without sacrificing the optimality of the sensor selection.

Although our approaches have general application, we present our algorithms in the context of a scenario in which a surface ship is tracked by an acoustic homing torpedo. Specifically, we consider the target-acquisition phase in which the torpedo uses electroacoustic transducers and passive beamforming to obtain bearing measurements from the target and estimate the target's position and velocity. In the acquisition phase, the torpedo must move slowly to minimize the acoustic interference at the torpedo's transducers [15]. The torpedo maneuvers relative to the target to improve the target observability. The objective of the sensor scheduling problem is thus to obtain a sequence of torpedo headings that minimize the predicted squared error in the target position estimate over a future time-horizon. As stated, this sensor scheduling problem has a continuous-valued configuration variable (the torpedo heading), and could potentially be addressed using stochastic approximation techniques (e.g., [6]). However, these techniques are extremely computationally demanding and often require careful tuning for successful application. As an alternative, we quantize the continuous-valued control variable into a finite set of possible headings and apply discrete optimization techniques over these values.

Our two proposed scheduling algorithms use different approximation techniques to predict the expected future cost. The first scheduling algorithm is a covariance-based (CB) algorithm which can be applied when the scheduler cost is a function of the state estimate error covariance matrix. The second algorithm is an unscented transform-based (UTB) algorithm that uses an unscented transform in conjunction with Monte Carlo sequential sampling to compute general costs (e.g., covariance-based costs or information-theoretic costs) that depend on the future system state and measurements. As we will demonstrate, the UTB algorithm performs better than the CB algorithm; however, the computational efficiency of the CB algorithm makes it an attractive choice for computationally constrained tracking systems.

To reduce the computational burden in nonmyopic scheduling with both the CB and the UTB algorithms, we propose the use of two B&B based pruning algorithms to efficiently obtain the optimal sensor sequence. We designed the first algorithm by combining a breadth-first search (BFS) and greedy search (GS) with the B&B method. The second algorithm is a uniform-cost search (UCS) B&B algorithm. The UCS-based pruning algorithm is more efficient in terms of processing time, while the BFS-GS algorithm is better in memory usage.

This paper is organized as follows. In Section 2, we formulate the tracking scenario and describe the tracking algorithm. In Section 3, we present the optimization framework for sensor scheduling, and propose the two sensor scheduling algorithms for nonmyopic scheduling. In Section 4, we discuss the two optimal pruning algorithms, and in Section 5, we demonstrate the improved performance of our algorithms using Monte Carlo methods. Note that our adopted notation is summarized in Table 1.

2. TARGET TRACKING SCENARIO

For the sake of concreteness, we formulate the sensor scheduling problem in the context of a scenario in which an acoustic homing torpedo tracks a surface target (Figure 1) [15]. Note however, that our scheduling algorithms can be readily adapted to other problems with discrete configuration options including tracking an airborne target with a missile or optimizing the tracking performance in a network of sensors where the target belief transfer (between two sensors) is constrained by network energy and bandwidth costs [16].

2.1. Problem formulation

We consider a target moving in two-dimensions. The target state at time k is $\mathbf{x}_k = [x_k \;\; \dot{x}_k \;\; y_k \;\; \dot{y}_k]^T$, where x_k and y_k represent the target position in Cartesian coordinates, and $\dot{x}_k$ and $\dot{y}_k$ represent the corresponding velocity. We model the target dynamics with a constant-velocity model given by

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}. \tag{1}$$

Here, $\mathbf{F}$ is the state transition matrix, and $\mathbf{w}_k$ is a zero-mean white Gaussian sequence with covariance $\mathbf{Q}$.

At each time k, the torpedo's acoustic sensors obtain the noisy bearing measurement z_k:

$$z_k = h(\mathbf{x}_k; x_k^s, y_k^s) + \nu_k \triangleq \tan^{-1}\left(\frac{y_k - y_k^s}{x_k - x_k^s}\right) + \nu_k, \tag{2}$$

where ν_k is zero-mean white Gaussian noise with variance σ^2, x_k^s and y_k^s denote the torpedo's x and y coordinates at time k, and $h(\mathbf{x}_k; x_k^s, y_k^s)$ is the measurement function. $Z_k \triangleq z_{1:k}$ denotes the sequence of sensor measurements from 1 to k.

TABLE 1: Adopted notation.

Notation	Definition
$\mathbf{x}_{0:k} \triangleq \mathbf{X}_k$	The state sequence from time 0 to k: $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k$
$z_{1:k} \triangleq Z_k$	The measurement sequence from time 1 to k: $z_1, z_2, \dots, z_k$
$z_{k+1:k+m} \triangleq \mathcal{Z}_{k+m}$	The measurement sequence from time $k+1$ to $k+m$
$s_{1:k} \triangleq S_k$	The configured sensor-position sequence from time 1 to k: $s_1, s_2, \dots, s_k$
$s_{k+1:k+m} \triangleq \mathcal{S}_{k+m}$	The configured sensor-position sequence from time $k+1$ to $k+m$
$\hat{\mathbf{x}}_{k\|k}$	State estimate at time k based on Z_k
$\hat{\mathbf{x}}_{k+m\|k}$	State estimate at time $k+m$ based on Z_k
$\overline{\mathbf{x}}_{k+m\|k+m}$	State estimate at time $k+m$ based on Z_k and $\mathcal{Z}_{k+m}$
$\mathbf{P}_{k\|k}$	Covariance matrix of estimate error at time k based on Z_k
$\hat{\mathbf{P}}_{k\|k}$	Approximate covariance matrix of estimate error at time k based on Z_k
$\hat{\mathbf{P}}_{k+m\|k}$	Approximate covariance matrix of estimate error at time $k+m$ based on Z_k
$\check{P}_{k+m\|k+r}$	Approximate covariance matrix of estimate error at time $k+m$ based on Z_k and the effect using sensor sequence $\mathcal{S}_{k+r}$, $1 \leq r \leq m$
$\overline{P}_{k+m\|k+r}$	Approximate covariance matrix of estimate error at time $k+m$ based on Z_k and $\mathcal{Z}_{k+r}$, $1 \leq r \leq m$

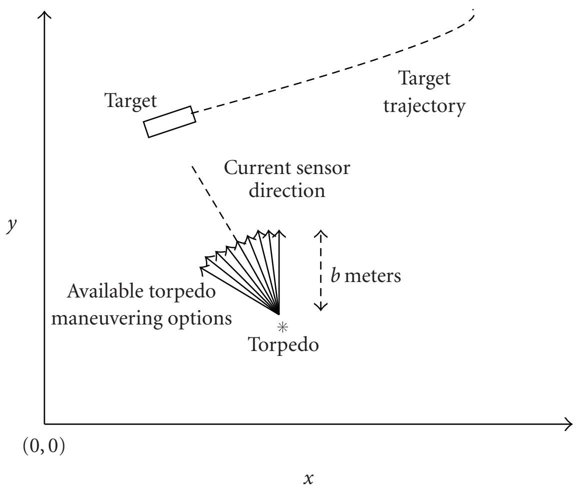


FIGURE 1: Tracking scenario: a sea target is tracked by a torpedo. At each time epoch, the torpedo can change heading by one of nine possible values and then move b meters.

At a given time k, the torpedo can change heading by one of the nine possible values $\{i\pi/16,\ i = -4, \dots, 4\}$ as shown in Figure 1; it then moves b meters along its new heading. These possible torpedo motions define the set of possible sensor configuration options for this problem; in the following, we will refer to these as sensor motion or sensor configuration options.

We denote the configured sensor position at time k by $s_k \triangleq (x_k^s, y_k^s)$, and the sequence of positions from 1 to k by $S_k \triangleq s_{1:k}$. The sensor configuration at k is denoted by g_k. We denote the set of allowable sensor configurations as $\mathcal{G}$ and the number of configurations as U. For example, in Figure 1, there are $U = 9$ allowable sensor configurations at each time k: move along the current heading or change to one of eight possible new directions. The configured sensor position s_{k+1} at time $k+1$ is a deterministic function of g_{k+1} and s_k; we assume that there is no uncertainty or error in the sensor movement. Thus, given the initial sensor position s_0 and the sequence of sensor configurations $g_1, \dots, g_k$, we can obtain the configured sensor position s_k at time k.

2.2. Target tracking using a particle filter

The extended Kalman filter is often not robust in bearing-only tracking because of target observability problems; for this reason, we use a particle filter to track the target [17]. In a particle filter, the posterior probability density $p(\mathbf{x}_k \mid Z_k, S_k)$ is approximated by N particles $\mathbf{x}_k^i$ and associated importance weights w_k^i, $i = 1, \dots, N$, as $p(\mathbf{x}_k \mid Z_k, S_k) \approx \sum_{i=1}^N w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i)$. The minimum mean-square error (MMSE) estimate of the target state is the mean $\hat{\mathbf{x}}_{k|k} = E_{\mathbf{x}_k|Z_k,S_k}[\mathbf{x}_k \mid Z_k, S_k] \approx \sum_{i=1}^N w_k^i \mathbf{x}_k^i$ of this density, where $E[\cdot]$ denotes expectation;[1] the covariance matrix of the estimate error is approximated as $\hat{\mathbf{P}}_{k|k} \approx \sum_{i=1}^N w_k^i (\mathbf{x}_k^i - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k^i - \hat{\mathbf{x}}_{k|k})^T$.

At each time k, the particles $\mathbf{x}_k^i$ are drawn from the prior density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$; after obtaining a measurement z_k, the weights are updated recursively using $w_k^i = w_{k-1}^i p(z_k \mid \mathbf{x}_k^i, s_k) / (\sum_{j=1}^N w_{k-1}^j p(z_k \mid \mathbf{x}_k^j, s_k))$. Resampling is performed to avoid degeneracy of the particles [17].

3. NONMYOPIC SENSOR SCHEDULING

Nonmyopic scheduling is important when myopic decisions result in poor estimation performance. In the current

[1] Note that when necessary, we use the notation $E_x[\cdot]$ to clarify that the expectation is with respect to the density of x.

tracking scenario, the need for nonmyopic scheduling arises due to the constrained maneuverability of the sensor. Nonmyopic sensor scheduling can be realized in two ways. The first is open-loop (OL) scheduling, in which the scheduling is performed only after all multistep decisions are exhausted [18]. The second is open-loop feedback (OLF) scheduling, in which only the first scheduling decision is executed, and the nonmyopic scheduling is repeated at each time step [18–22]. Although our algorithm description is based on OL scheduling, the optimization framework for both scheduling schemes is the same [18]. We will demonstrate through our results that OLF scheduling is better than OL scheduling due to the feedback obtained in scheduling decisions at each time step. Next, we describe the optimization framework before presenting our new sensor scheduling algorithms.

3.1. Optimization framework

Following the scenario in Figure 1, the sensor can be configured in U distinct ways at each time step k. At any given time k, our objective is to obtain the best sensor-configuration sequence over the next M time-steps in order to minimize the scheduler cost. We denote a sensor-configuration sequence by an M-tuple: $\mathcal{S}_{k+M} \triangleq [s_{k+1} \ \ s_{k+2} \ \ \cdots \ \ s_{k+M}]^T$, where s_{k+m} is the configured sensor position at time $k+m$ (m steps in the future). Note that there is a total of U^M distinct sensor sequences of length M.

We denote the scheduler cost at time $k+m$ by $J(s_{k+m})$. We define the total scheduler cost for a particular sensor sequence $\mathcal{S}_{k+M}$ as

$$J(\mathcal{S}_{k+M}) = \sum_{m=1}^{M} J(s_{k+m}). \tag{3}$$

We seek the optimal sequence $\mathcal{S}_{k+M}^{\text{opt}}$ that minimizes $J(\mathcal{S}_{k+M})$:

$$\mathcal{S}_{k+M}^{\text{opt}} = \arg\min_{\mathcal{S}_{k+M}} \{J(\mathcal{S}_{k+M})\}. \tag{4}$$

Equation (4) is a discrete optimization problem, where the scheduler cost is optimized over the finite set of possible sensor sequences. Note that our rationale for using the additive scheduler-cost structure[2] in (3) is that the costs in this paper are both stochastic and predictive; the scheduler costs are obtained by computing an expectation over the predicted state distribution. As M increases, the accuracy with which tracking performance can be predicted decreases. Thus, we do not rely only on the terminal cost of a sensor sequence, but also on the costs at intermediate points in time.

We consider two different scheduler costs $J(s_{k+m})$ in this paper. The first is the determinant of the predicted state estimate error covariance matrix at time $k+m$. Specifically with $\mathcal{Z}_{k+m} \triangleq z_{k+1:k+m}$,

$$\begin{aligned} J(s_{k+m}) &= |\mathbf{P}(s_{k+m})| \\ &= \left| E_{\mathbf{x}_{k+m},\mathcal{Z}_{k+m}}\left[(\mathbf{x}_{k+m} - \hat{\mathbf{x}}_{k+m|k+m})(\mathbf{x}_{k+m} - \hat{\mathbf{x}}_{k+m|k+m})^T \right] \right|. \end{aligned} \tag{5}$$

Minimizing this cost implies reducing the volume of the covariance ellipsoid [23].

The second cost is the Kullback-Leibler (KL) distance between the approximate predicted and filtered state densities. This is an information-theoretic metric that can be used to measure the average information gain in using each sensing action [24–26]. The KL distance cost is defined as $J(s_{k+m}) = E_{\mathcal{Z}_{k+m}|s_{k+m}}[\mathcal{C}(\mathcal{Z}_{k+m}, s_{k+m})]$, where $\mathcal{C}(\mathcal{Z}_{k+m}, s_{k+m})$ is a conditional cost function [27]:

$$\begin{aligned} &\mathcal{C}(\mathcal{Z}_{k+m}, s_{k+m}) \\ &\quad = -\int_{\mathbf{x}_{k+m}} \hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}_{k+m}, \mathcal{S}_{k+m}) \\ &\qquad \times \log\left[\frac{\hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}_{k+m}, \mathcal{S}_{k+m})}{\hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}_{k+m-1}, \mathcal{S}_{k+m-1})} \right] d\mathbf{x}_{k+m}. \end{aligned} \tag{6}$$

Here, $\hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}_{k+m-1}, \mathcal{S}_{k+m-1})$ and $\hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}_{k+m}, \mathcal{S}_{k+m})$ are approximations of the predicted and filtered densities at time $k+m$. Note the negative sign in (6); minimizing the conditional cost maximizes the KL distance, as desired.

The determinant cost approximates the target uncertainty using only the first- and second-order statistics of the posterior distribution. This cost can be approximately computed efficiently using the recursive Riccati equation, as implemented by the CB algorithm in Section 3.2.1. The KL distance cost depends on the entire posterior distribution and directly measures the average information contributed by each sensor configuration about the target state. However, the KL distance cost is computationally more expensive than the determinant cost as the KL distance cost cannot be computed using closed-form Riccati-like recursive formulations.

3.2. Proposed nonmyopic scheduling algorithms

We propose two nonmyopic sensor scheduling algorithms: the CB algorithm and the UTB algorithm. Both algorithms find the optimal sequence of sensor uses by searching exhaustively over all possible sequences. In principle, this requires the computation of $J(\mathcal{S}_{k+m})$ for each possible sequence $\mathcal{S}_{k+m}$. We note that for any two sequences, $\mathcal{S}_{k+m+1}^1$ and $\mathcal{S}_{k+m+1}^2$ ($1 \le m < M$), that have the same initial subsequence $\mathcal{S}_{k+m}$, the computation of $J(\mathcal{S}_{k+m})$ is redundant when concurrently computing $J(\mathcal{S}_{k+m+1}^1)$ and $J(\mathcal{S}_{k+m+1}^2)$; this redundancy could

[2] Note that in the current application scenario, both additive scheduler-cost in (3) and terminal scheduler-cost (in which we minimize $J(\mathcal{S}_{k+M})$ to obtain the best sensor sequence) resulted in similar tracking performance.

```
For each possible sequence of sensors $\mathcal{S}_{k+M} = s_{k+1:k+M}$
    (1) Initialize: $\hat{\mathbf{x}}_{k|k} = \sum_{i=1}^{N} w_k^i \mathbf{x}_k^i$, $\check{\mathbf{P}}_{k|k} = \hat{\mathbf{P}}_{k|k} = \sum_{i=1}^{N} w_k^i (\mathbf{x}_k^i - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k^i - \hat{\mathbf{x}}_{k|k})^T$
    (2) For $m$=1 to $M$,
    – Project the state estimate and covariance matrix of estimate error:
        (i)   $\hat{\mathbf{x}}_{k+m|k} = \mathbf{F}\hat{\mathbf{x}}_{k+m-1|k}$   (7)
        (ii)  $\check{\mathbf{P}}_{k+m|k+m-1} = \mathbf{F}\check{\mathbf{P}}_{k+m-1|k+m-1}\mathbf{F}^T + \mathbf{Q}$   (8)
    – Compute the Jacobian matrix $\mathbf{H}_{k+m}$:
        (iii) $\mathbf{H}_{k+m} = \left[\frac{\partial\theta}{\partial x}\ \frac{\partial\theta}{\partial \dot{x}}\ \frac{\partial\theta}{\partial y}\ \frac{\partial\theta}{\partial \dot{y}}\right]^T \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{k+m|k}}$ where $\theta = h(\mathbf{x}; x^s, y^s)$   (9)
    – Update the predicted covariance matrix of estimate error:
        (iv)  $\check{\mathbf{P}}_{k+m|k+m} = \left[\check{\mathbf{P}}_{k+m|k+m-1}^{-1} + \frac{1}{\sigma^2}\mathbf{H}_{k+m}\mathbf{H}_{k+m}^T\right]^{-1}$   (10)
    – Calculate $J(s_{k+m}) = |\check{\mathbf{P}}_{k+m|k+m}|$
        End
    (3) Calculate $J(\mathcal{S}_{k+M})$ using (3)
End
Choose the optimal sequence of sensors using (4)
```

Algorithm 1: The CB algorithm.

be easily eliminated in the actual implementation of the algorithm.

3.2.1. Covariance-based sensor scheduling

The covariance-based (CB) sensor scheduling algorithm uses the covariance-based cost and is particularly well-suited for tracking systems with limited computational and memory resources [28]. Specifically, the computational complexity of the CB algorithm in obtaining $J(s_{k+m})$ for a given s_{k+m} is in the order of $O(n_x^3)$, where n_x is the dimension of $\mathbf{x}_k$.

In the CB algorithm, we approximate $\mathbf{P}(s_{k+m})$ in (5) by linearizing the measurement model in (2) about a predicted target state $\hat{\mathbf{x}}_{k+m|k}$; we denote this approximation by $\check{\mathbf{P}}_{k+m|k+m}$. Our iterative CB algorithm is summarized in Algorithm 1. It is initialized by the estimates $\hat{\mathbf{x}}_{k|k}$ and $\hat{\mathbf{P}}_{k|k}$ computed at time k by a particle filter (in Section 2.2). For each sequence $\mathcal{S}_{k+m}$, equations (i) and (ii) of Algorithm 1 are used to predict $\hat{\mathbf{x}}_{k+m|k}$ and $\check{\mathbf{P}}_{k+m|k+m-1}$ to time $k+m$; we then linearize $h(\mathbf{x}; x^s, y^s)$ about $\hat{\mathbf{x}}_{k+m|k}$ to compute the Jacobian matrix $\mathbf{H}_{k+m}$ in equation (iii). $\check{\mathbf{P}}_{k+m|k+m}$ is obtained using equation (iv) in Algorithm 1; the determinant scheduler cost is then obtained as $J(s_{k+m}) = |\check{\mathbf{P}}_{k+m|k+m}|$. Finally, $J(\mathcal{S}_{k+M})$ is obtained using (3). Note that equations (i) and (ii) of Algorithm 1 correspond to the prediction step of the extended Kalman filter (EKF), while equation (iv) of Algorithm 1 corresponds to the update step of the EKF.

The CB is similar to the PCRLB algorithm in [8], but was developed independently [28]. The two algorithms differ in the calculation of the sensor information term, that is, $(1/\sigma^2)\mathbf{H}_{k+m}\mathbf{H}_{k+m}^T$; while the CB algorithm computes it using the predicted state estimate $\hat{\mathbf{x}}_{k+m|k}$, the PCRLB algorithm computes an expected value of the sensor information term using the predicted state density $p(\mathbf{x}_{k+m} \mid Z_k, S_k)$.

3.2.2. Unscented transform-based sensor scheduling

The motivation for the unscented transform-based (UTB) algorithm is to provide a generalized framework that allows sensor scheduling using information-theoretic costs. The UTB algorithm does not require the Jacobian matrix; this is useful when it is not possible to obtain the Jacobian matrix analytically. For instance, in a tracking scenario where the measurements are binary valued (detect or no-detect) and depend probabilistically on the state (e.g., through a probability of detection), it is not possible to obtain an expression of the Jacobian matrix.

In the UTB algorithm, the key idea is to sample future state and measurement particles, and to calculate expected costs using these particles. We first investigated sequential sampling methods [3], where the particles for future states and measurements were obtained directly using the particle filter. These methods were computationally too expensive.

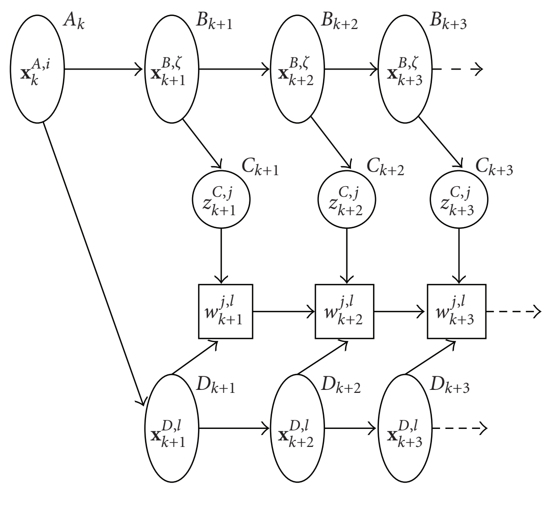


FIGURE 2: Sets of particles used to compute the expected future cost for the UTB algorithm.

Grid-based sampling techniques as used in [25, 29] are also computationally expensive as they require large number of particles to compute expected scheduler costs. Instead, we propose in this paper to use the unscented transform (UT) to generate future particles [30]. As these sample particles are few in number, the computational load in calculating the scheduler costs is significantly reduced.

The UTB algorithm is summarized in Algorithm 2. In this algorithm, we use several sets of particles as shown in Figure 2. At time $k+m$, the particle sets used are $B_{k+m} = \{\mathbf{x}_{k+m}^{B,\zeta}\}$, $\zeta = 1,\dots,\mathcal{N}_\sigma$, which is a predicted set of $\mathcal{N}_\sigma$ state particles calculated using the UT (where $\mathcal{N}_\sigma$ is the number of sigma points obtained using the UT) and approximates $p(\mathbf{x}_{k+m} \mid \mathbf{x}_k, S_k)$; $C_{k+m}^{s_{k+m}} = \{z_{k+m}^{C,j}\}$, $j = 1,\dots,E$, which is a predicted set of E measurement[3] particles calculated using the $\mathcal{N}_\sigma$ state particles and approximates $p(z_{k+m} \mid \mathbf{x}_k, s_{k+m})$; and $D_{k+m} = \{\mathbf{x}_{k+m}^{D,l}\}$, $l = 1,\dots,L$, which is a predicted set of L ($\le N$) state particles and approximates $p(\mathbf{x}_{k+m} \mid \mathbf{x}_k, S_k)$. Also, $\mathcal{X}_{k+m}^{D,l} \triangleq [\mathbf{x}_{k+1}^{D,l} \ \cdots \ \mathbf{x}_{k+m}^{D,l}]^T$, $l = 1,\dots,L$, and $\mathcal{Z}_{k+m}^{C,j} \triangleq [z_{k+1}^{C,j} \ \cdots \ z_{k+m}^{C,j}]^T$, $j = 1,\dots,E$, are defined as the lth predicted state sequence and the jth predicted measurement sequence, respectively, from time $k+1$ to $k+m$. We now describe the M-step UTB algorithm.

Initialize $A_k = \{\mathbf{x}_k^{A,i}\}$, $i = 1,\dots,N$, as the set of resampled particles computed by the particle filter at time k.

Initialize $D_{k+1} = \{\mathbf{x}_{k+1}^{D,l}\}$, $l = 1,\dots,L$, by randomly sampling L particles from the set A_k, and predicting these particles to $k+1$ by sampling from the distribution $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k^{A,l})$. Initialize the set $B_{k+1} = \{\mathbf{x}_{k+1}^{B,\zeta}\}$, $\zeta = 1,\dots,\mathcal{N}_\sigma$, by performing a UT on the set A_k through the steps (i) to (iii) in the following.

[3] We use s_{k+m} as a superscript in $C_{k+m}^{s_{k+m}}$ to denote the explicit dependence of the measurement set on the sensor s_{k+m}.

(i) Compute the predicted mean and predicted covariance matrix of estimate error at time k:

$$\overline{\mathbf{x}}_{k|k} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_k^{A,i},$$
$$\overline{\mathbf{P}}_{k|k} = \frac{1}{N}\sum_{i=1}^{N} \left(\mathbf{x}_k^{A,i} - \overline{\mathbf{x}}_{k|k}\right)\left(\mathbf{x}_k^{A,i} - \overline{\mathbf{x}}_{k|k}\right)^T. \tag{15}$$

(ii) Define $\mathbf{x}_{k|k}^a = [\overline{\mathbf{x}}_{k|k}^T \ \mathbf{0} \ \mathbf{0}]^T$ as a concatenation of the state, process noise, and measurement noise vectors, and $\mathbf{P}_{k|k}^a = \operatorname{diag}(\overline{\mathbf{P}}_{k|k}, \mathbf{Q}, \sigma^2)$ as the covariance of $\mathbf{x}_{k|k}^a$. The length of the vector $\mathbf{x}_{k|k}^a$ is denoted by $n_a = 9$.

(iii) Using the UT [31], we deterministically compute $\mathcal{N}_\sigma = 2n_a + 1$ sigma points from A_k. The sigma points are defined as $X_k^\zeta \triangleq [X_k^{\mathbf{x},\zeta} \ X_k^{\mathbf{w},\zeta} \ X_k^{\mathbf{v},\zeta}]^T$, $\zeta = 1,\dots,\mathcal{N}_\sigma$, and are computed as [31]

$$X_k^\zeta = \begin{cases} \mathbf{x}_{k|k}^a, & \zeta = 0, \\ \mathbf{x}_{k|k}^a + \Lambda_\zeta, & \zeta = 1,\dots,\dfrac{\mathcal{N}_\sigma - 1}{2}, \\ \mathbf{x}_{k|k}^a - \Lambda_{\zeta - n_a}, & \zeta = \dfrac{\mathcal{N}_\sigma + 1}{2},\dots,\mathcal{N}_\sigma. \end{cases} \tag{16}$$

Here $X_k^{\mathbf{x},\zeta}$, $X_k^{\mathbf{w},\zeta}$, and $X_k^{\mathbf{v},\zeta}$ denote the partition of X_k^ζ in the target-state space, process-noise space, and measurement-noise space, respectively. Furthermore, Λ_ζ is the ζth column of Λ, $\Lambda = \sqrt{(n_a+\lambda)\mathbf{P}_{k|k}^a}$, and $\lambda = a_0^2(n_a+\kappa) - n_a$. Note that $0 \le a_0 \le 1$ determines the spread of the sigma points around $\mathbf{x}_{k|k}^a$. A value of $a_0 = 0.1$ was chosen through experimentation to ensure that the sigma points are neither spaced too far from the mean nor too close to the mean. The secondary scaling parameter κ is generally set to zero [31]. Now, using the sigma points, we calculate the elements of the sets B_{k+1} as $\mathbf{x}_{k+1}^{B,\zeta} = \mathbf{F}X_k^{\mathbf{x},\zeta} + X_k^{\mathbf{w},\zeta}$, $\zeta = 1,\dots,\mathcal{N}_\sigma$.

We then iterate the following steps for $m = 1$ to M.

Step 1. For $m > 1$, obtain the elements of D_{k+m} as $\mathbf{x}_{k+m}^{D,l} = \mathbf{F}\mathbf{x}_{k+m-1}^{D,l} + \boldsymbol{\xi}_D$, $l = 1,\dots,L$, where $\boldsymbol{\xi}_D$ is a random sample drawn from a Gaussian distribution of zero mean and covariance $\mathbf{Q}$.

Step 2. For $m > 1$, obtain the elements of B_{k+m} as $\mathbf{x}_{k+m}^{B,\zeta} = \mathbf{F}\mathbf{x}_{k+m-1}^{B,\zeta} + \boldsymbol{\xi}_B$, $\zeta = 1,\dots,\mathcal{N}_\sigma$, where $\boldsymbol{\xi}_B$ is a random sample drawn from a Gaussian distribution of zero mean and covariance $\mathbf{Q}$.

Step 3. Obtain η measurements for each sigma point in B_{k+m} using the distribution $p(z_{k+m} \mid X_{k+m-1}^{\mathbf{x},\zeta}, s_{k+m})$ to form the measurement set $C_{k+m}^{s_{k+m}}$ with $E = \eta\mathcal{N}_\sigma$ measurement particles.

Step 4. Using the sets $C_{k+m}^{s_{k+m}}$ and D_{k+m}, we compute the scheduler cost $J(s_{k+m})$ at time $k+m$ using equations (ii)–(v)

```
For each possible sequence of sensors $\mathcal{S}_{k+M}$
    (1) Initialize: $A_k$, $B_{k+1}$ and $D_{k+1}$
    (2) For $m = 1$ to $M$,
    – Obtain sets $B_{k+m}$, $C_{k+m}^{s_{k+m}}$, and $D_{k+m}$ using Steps 1–3 in Section 3.2.2
    – Compute the cost $J(s_{k+m})$:
        (i) Compute $w_{k+m}^{j,l}$ using the particles in $D_{k+m}$ and $C_{k+m}^{s_{k+m}}$:
            $w_{k+m}^{j,l} \propto p(z_{k+m}^{j} \mid \mathbf{x}_{k+m}^{l}, s_{k+m}) p(\mathcal{Z}_{k+m-1}^{j} \mid \mathcal{X}_{k+m-1}^{l}, \mathcal{S}_{k+m-1})$   (11)
        (ii) Compute the approximate conditional cost function $\mathcal{C}(\mathcal{Z}_{k+m}^{(j)}, s_{k+m})$ in (17) and (18) using $w_{k+m}^{j,l}$ and $\mathbf{x}_{k+m}^{l}$.
        (iii) Compute the approximate conditional density of $\mathcal{Z}_{k+m}^{j}$ using $D_{k+m}$:
            $\widehat{p}(\mathcal{Z}_{k+m}^{j} \mid Z_k, S_{k+m}) \approx \sum_{l=1}^{L} \widehat{p}(\mathcal{Z}_{k+m}^{j} \mid \mathbf{x}_{k+m}^{l}, S_{k+m}) = \sum_{l=1}^{L} p(z_{k+m}^{j} \mid \mathbf{x}_{k+m}^{l}, s_{k+m}) p(\mathcal{Z}_{k+m-1}^{j} \mid \mathcal{X}_{k+m-1}^{l}, \mathcal{S}_{k+m-1})$   (12)
        (iv) Compute the expectation of $\mathcal{C}(\mathcal{Z}_{k+m}^{j}, s_{k+m})$ at time $k+m$ as
            $E_{\mathcal{Z}_{k+m}}[\mathcal{C}(\mathcal{Z}_{k+m}^{j}, s_{k+m})] \approx \frac{\sum_{j=1}^{E} \gamma_{k+m}^{j} \mathcal{C}(Z_{k+m}^{j}, s_{k+m})}{\sum_{j=1}^{E} \gamma_{k+m}^{j}}$, where $\gamma_{k+m}^{j} \triangleq \widehat{p}(Z_{k+m}^{j} \mid Z_k, S_{k+m})$   (13)
        (v) Compute the scheduler cost at time $k+m$ as
            $J(s_{k+m}) = \begin{cases} E_{\mathcal{Z}_{k+m}}[\mathcal{C}(\mathcal{Z}_{k+m}^{j}, s_{k+m})] & \text{for KL cost} \\ |\mathbf{P}(s_{k+m})| \triangleq |E_{\mathcal{Z}_{k+m}}[\mathcal{C}(\mathcal{Z}_{k+m}^{j}, s_{k+m})]| & \text{for determinant cost.} \end{cases}$   (14)
    (3) Calculate the total scheduler cost using (3)
End
Choose the optimal sequence of sensors using (4)
```

Algorithm 2: The UTB algorithm.

in Algorithm 2. We then obtain the total scheduling cost $J(\mathcal{S}_{k+m})$ using (3); optimizing over all sequences gives the optimal sensor sequence $\mathcal{S}_{k+m}^{\text{opt}}$ using (4). Note that when possible in Algorithm 2 and hereafter, we drop the superscript C from $\mathcal{Z}_{k+m}^{C,j}$ and the subscript D from $\mathcal{X}_{k+m}^{D,l}$ and $\mathbf{x}_{k+m}^{D,l}$, to simplify the notation.

The method in Algorithm 2 can be used for any conditional cost function that depends on future measurements. The conditional cost function for covariance-based costs is given as

$$\mathcal{C}_{\text{COV}}(\mathcal{Z}_{k+m}^{j}, s_{k+m}) = \sum_{l=1}^{L} w_{k+m}^{j,l} \left(\mathbf{x}_{k+m}^{l} - \overline{\mathbf{x}}_{k+m}^{j}\right)\left(\mathbf{x}_{k+m}^{l} - \overline{\mathbf{x}}_{k+m}^{j}\right)^{T}, \tag{17}$$

where $\overline{\mathbf{x}}_{k+m}^{j} = \sum_{l=1}^{L} w_{k+m}^{j,l} \mathbf{x}_{k+m}^{l}$, and $w_{k+m}^{j,l}$ are the weights obtained in step (ii) of Algorithm 2.

For the KL distance cost, the corresponding conditional cost is derived in Appendix A, and is given by

$$\mathcal{C}_{\text{KL}}(\mathcal{Z}_{k+m}^{j}, s_{k+m}) = \sum_{l=1}^{L} -w_{k+m}^{j,l} \log\left[\frac{w_{k+m}^{j,l}}{w_{k+m-1}^{j,l}}\right]. \tag{18}$$

Equation (18) resembles the KL distance between two discrete distributions and can be interpreted in a similar way. The particles in set D each has weights equal to $w_{k+m-1}^{j,l}$, and represent our belief of the future state. Each predicted measurement z_{k+m}^{j} updates these weights to $w_{k+m}^{j,l}$, according to the measurement model. The gain in information for each predicted measurement is calculated using (18), which is then averaged with respect to the measurement density $p(\mathcal{Z}_{k+m}^{j} \mid Z_k, \mathcal{S}_{k+m})$.

It must be noted that equation (i) (derived in Appendix B) in Algorithm 2 allows us to incorporate the effect of

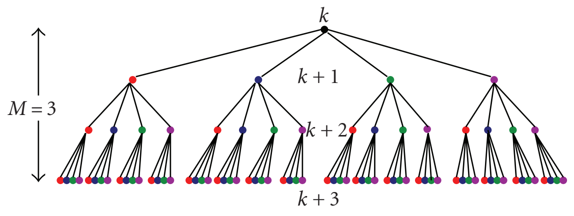


Figure 3: An illustrative configuration tree with $U = 4$ configuration choices and a time horizon of $M = 3$.

predicted measurements z^j_{k+m} on the predicted density function $p(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1})$. Although L and E are required to be large numbers in order to accurately predict the scheduler costs, this results in a significant increase in computational complexity. Furthermore, as we are mainly interested in the relative tracking performance achievable with the available sensor configurations, we can trade off the computational cost of scheduling with the accuracy of the predicted tracking performance. To this effect, we choose $L = 2000$ and $E = 380$ ($\eta = 20$) for the state and measurement particles.

We further note that in order to compute $w^{j,l}_{k+m}$, we need to store $p(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m-1}, \mathcal{S}_{k+m-1})$ (equation (i) of Algorithm 2) in memory and access it only when required. However, as storing $p(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m-1}, \mathcal{S}_{k+m-1})$ requires a lot of memory, in this work the scheduler stores only the predicted measurements for each sensor configuration. We note that $p(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m-1}, \mathcal{S}_{k+m-1})$ is generated only once when concurrently computing $J(s_{k+m})$ for two sensor sequences having identical measurement history up to time $k+m-1$.

The computational complexity of the UTB algorithm in obtaining $J(s_{k+m})$ with the KL cost for a given s_{k+m} is in the order of $O(n_x EL)$; thus, the UTB algorithm is computationally more expensive than the CB algorithm. Furthermore, the computational complexity in obtaining $\mathbf{P}(s_{k+m})$ for the determinant cost in equation (v) of Algorithm 2, given the weights $w^{j,l}_{k+m}$ and γ^j_{k+m} (in equations (i) and (iv), resp., of Algorithm 2), is in the order of $O(n_x(n_x+2)EL)$. An alternative formulation in obtaining $\mathbf{P}(s_{k+m})$ is

$$\mathbf{P}(s_{k+m}) = \mathbf{P}_1(s_{k+m}) - \mathbf{P}_2(s_{k+m}), \qquad (19)$$

where $\mathbf{P}_1(s_{k+m}) = \sum_{l=1}^{L} \tilde{w}^l_{k+m}(\mathbf{x}^l_{k+m} - \overline{\mathbf{x}}_{k+m})(\mathbf{x}^l_{k+m} - \overline{\mathbf{x}}_{k+m})^T$ with

$$\overline{\mathbf{x}}_{k+m} = \sum_{l=1}^{L} \tilde{w}^l_{k+m}\mathbf{x}^l_{k+m},$$

$$\tilde{w}^l_{k+m} = \frac{\sum_{j=1}^{E} p(z^j_{k+m} \mid \mathbf{x}^l_{k+m}, s_{k+m})\, p(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m-1}, \mathcal{S}_{k+m-1})}{\sum_{j=1}^{E}\sum_{l=1}^{L} p(z^j_{k+m} \mid \mathbf{x}^l_{k+m}, s_{k+m})\, p(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m-1}, \mathcal{S}_{k+m-1})}, \quad l = 1,\dots,L, \qquad (20)$$

$$\mathbf{P}_2(s_{k+m}) = \frac{\sum_{j=1}^{E} \gamma^j_{k+m}(\overline{\mathbf{x}}^j_{k+m} - \overline{\mathbf{x}}_{k+m})(\overline{\mathbf{x}}^j_{k+m} - \overline{\mathbf{x}}_{k+m})^T}{\sum_{j=1}^{E} \gamma^j_{k+m}}.$$

This formulation avoids computing $\mathcal{C}_{\mathrm{COV}}(\mathcal{Z}^j_{k+m}, s_{k+m})$ (in (17)) E times for equation (ii) in Algorithm 2, and it reduces the computational complexity in obtaining $\mathbf{P}(s_{k+m})$ to the order of $O(n_x EL)$, that is, by an order of $n_x + 2 = 6$.

4. PRUNING ALGORITHMS FOR NONMYOPIC SENSOR SCHEDULING

4.1. Tree search and pruning algorithms

The sensor sequences (of length M) can be arranged in a tree of depth M as shown in Figure 3, with each depth-m node of the tree depicting a configured sensor position at time $k+m$. Thus, the sensor scheduling problem can be posed as a tree search problem, where the best sensor sequence corresponds to the lowest-cost branch of this tree.

We use the following terminology. A node is *open* if its cost has been computed, *expanded* if all its children have been opened, and *pruned* if the node and its children have been removed from the tree. Note that during a node expansion, we compute the cost of all of the children nodes. Pruning a node with optimality means that the pruned node is guaranteed not to be a part of the best sensor sequence.

We implement the scheduling algorithms using different combinations of three search techniques: breadth-first search (BFS), uniform-cost search (UCS), and greedy search (GS). BFS expands the nodes in depth order; a depth-m node ($m > 1$) is expanded only when all shallower nodes have been expanded [11]. Algorithm 3 shows the pseudocode for BFS. BFS uses a list to store all the unexpanded nodes; newly opened nodes are always appended to the end of the list. Each level of the tree must be stored to generate the next level. The worst-case memory requirement (proportional to the maximum number of stored nodes) is $O(U^M)$. In addition, the worst case time complexity is also $O(U^M)$ [11].

```
Initialize: SolutionFound = FALSE and list = root node
While (SolutionFound = FALSE) and (there is a node in the list)
    Remove the first node from the list and expand it
    If depth of children nodes $\neq M$
        Sort the children nodes in ascending order of costs
        Append the sorted children nodes to the list
    else
        If solution is found
            Set SolutionFound = TRUE
        End
    end
end
```

Algorithm 3: Pseudocode for breadth-first search.

```
Initialize: SolutionFound = FALSE and list = root node
While (SolutionFound = FALSE) and (there is a node in the list)
    Expand the first node and remove it from the list
    If depth of children nodes $\neq M$
        Sort the children nodes in an ascending order of costs
        Prepend the list with sorted children nodes
    else
        Choose lowest cost depth $M$ open node as the best solution
        Set SolutionFound = TRUE
    end
end
```

Algorithm 4: Pseudocode for greedy search.

In the UCS, the lowest-cost unexpanded node of a tree is expanded regardless of its depth in the tree [11]. The pseudocode for UCS is exactly the same as that for BFS, except that instead of appending the sorted children nodes to the list, we insert the children nodes into the list such that the updated list is in ascending order of cost. UCS is more time-efficient than BFS, but has the same memory complexity as BFS [11].

GS always expands the lowest-cost, lowest-depth, open node of the tree; Algorithm 4 shows its pseudocode. GS expands only the lowest-cost open node at each depth of the tree, so its memory and time complexity is $O(UM)$. GS does not search the tree exhaustively and does not guarantee the optimal solution.

With exhaustive search, a total of U^M sensor sequences must be considered to obtain the optimal sensor sequence. As M increases, the number of sensor sequences grows exponentially; since the computational time and memory usage increase exponentially as well, it is imperative to reduce the search space as much as possible. We propose two optimal pruning algorithms that significantly reduce the computational burden in obtaining the sensor sequences. The pruning algorithms are optimal as they provide the same best sensor sequence as the one obtained using an exhaustive search [32].

These pruning algorithms use the branch-and-bound technique; the B&B technique is often used to prune the search tree for problems such as the traveling-salesman problem, vehicle routing, and production planning [33, 34]. Application of this technique requires that lower bounds on the costs of all nodes in the tree are easier to compute than the actual costs of the nodes. Typically, in a B&B aided tree search, the tree is traversed using a search technique with desired time/memory tradeoffs; whenever a potential best solution is obtained, its cost is compared to the lower bounds of all the unexpanded open nodes. Any node whose lower bound is larger than the cost of the current best solution is pruned from the tree. B&B can significantly reduce the computational and memory requirements but typically does not eliminate exponential complexity. As part of our future work, we will investigate efficient search algorithms that do not require a complete enumeration of the search space.

4.2. Branch-and-bound-based pruning algorithms

We present two B&B based pruning algorithms in this section. The first pruning algorithm that we developed combines BFS and GS with the B&B technique, and is relatively efficient in memory usage. We call this the BFS-GS pruning algorithm. The second pruning algorithm is referred to as a best-first B&B algorithm [35] in the literature; it combines UCS with the B&B technique and is relatively efficient in processing time.

The pruning algorithms address two main issues of an exhaustive search: (a) each node expansion requires computation of the scheduler cost since the costs are stochastic in nature and are not known a priori, and (b) each open node (except depth-M nodes) requires memory to store the predicted state information. Specifically, for the CB algorithm, each node stores a mean vector and a covariance matrix, while for the UTB algorithm, each node stores a set of measurement particles. Additionally for each node, its cost, its status (open, close, or pruned), and an index to identify its position in the tree must be stored.

In simulations, we observed that the cost of some depth M nodes that resulted in improved tracking performance was lower than the cost of many intermediate depth nodes that resulted in poor performance. Furthermore, it was found that suboptimal techniques that accept the first candidate solution found (such as a pure GS or a combination of BFS and pure GS) yield poor tracking performance in comparison to an optimal search. This motivated us to use the B&B framework. The additive cost in (3) guarantees that for non-negative scheduler costs, any children of these poor performance intermediate depth nodes will have larger costs than the depth M nodes. Making use of this fact, we assign the lower bound on the cost of any unopened node as the cost of its nearest open ancestor. Specifically, for a given sensor sequence $\mathcal{S}_{k+m}$ with $m > 1$, the lower bound on $J(s_{k+m})$ is chosen as $J(s_{k+r})$, where s_{k+r} $(1 \leq r < m)$ corresponds to the deepest open node in $\mathcal{S}_{k+m}$. This bound is a valid lower bound because the additive cost structure in (3) guarantees that $J(s_{k+r}) \leq J(s_{k+m})$ for $r < m$. Although this bound is conservative, it works very well for our problem as demonstrated by our results in Section 5.3.2.

```
Initialize:    $c_{\min} = \infty$
Perform BFS up to depth $d_{\text{int}} < M$
Store the depth $d_{\text{int}}$ nodes in a list, sorted in ascending order
of cost
While there is a node in the list
    Expand the first node and remove it from the list
    If depth of children nodes = $M$
        If the lowest-cost child node has cost lower than $c_{\min}$
                Set $c_{\min}$ to this cost
                Set BestNode to this child
        end
    else
        Sort the children nodes in ascending order of costs
        Prepend the list with sorted children nodes
    end
    For all nodes in the list
        If cost of a node $\geq c_{\min}$
            remove the node from the list
        end
end
Trace back the BestNode to the root node to obtain $\mathcal{S}_{k+M}^{\text{opt}}$
```

Algorithm 5: Pseudocode for the BFS-GS pruning algorithm.

It must be noted that our B&B algorithms are applicable only with positive scheduler costs (e.g., determinant and trace of covariance matrix of estimate error, and entropy of the posterior distribution). Since the KL distance cost in (18) is negative, our B&B pruning algorithms cannot be used with the KL-based scheduling.

We now present our two pruning algorithms.

4.2.1. BFS-GS pruning algorithm

The pseudocode for our proposed BFS-GS pruning algorithm is provided in Algorithm 5. In this algorithm, we first perform a BFS to an intermediate depth d_{int}, and then beginning with the best node of depth d_{int}, we perform a GS to the terminating depth M. The GS gives an initial candidate path ending in a node with cost that we denote $c_{\min}$. We then repeat the following until there are no unexpanded open nodes.

Step 1. Compare the cost of all unexpanded open nodes to $c_{\min}$; prune any node whose cost is not less than $c_{\min}$. The additive cost guarantees that the best node cannot be a child of any pruned node.

Step 2. Perform a GS on the tree beginning at the lowest-cost open node; at each expansion compare the cost of the children nodes with $c_{\min}$ and prune away the nodes whose cost is not less than $c_{\min}$. If the GS gives a path with a terminal node whose cost is less than $c_{\min}$, set $c_{\min}$ to be this cost and the best path to be this path.

The intermediate depth d_{int} is an important factor for the BFS-GS pruning algorithm since the best node at this depth is used as a starting point for the GS to find an initial candidate solution. As d_{int} increases, the probability of the initial candidate solution being closer to the best solution increases. However, large values of d_{int} are undesirable because an exhaustive-search (here BFS) to depth d_{int} is conducted. At the same time, a small d_{int} is undesirable as the initial candidate solutions obtained using it are often of poor quality, which results in superfluous expansion of nodes. For the problem under consideration, we found that a good compromise for the BFS-GS algorithm is $d_{\text{int}} = \lceil M/2 \rceil$.

4.2.2. UCS pruning algorithm

The second pruning algorithm combines UCS with the B&B algorithm. In this algorithm, we first use a UCS to expand the nodes until the terminating depth M is reached. The lowest cost sensor sequence of length M is used as an initial candidate solution whose cost is denoted by $c_{\min}$. We then repeat the same two steps of the BFS-GS pruning algorithm, except that we use a UCS instead of the GS. The pseudocode for this algorithm is the same as that in Algorithm 5, except that we set $d_{\text{int}} = 1$ and instead of sorting the children nodes and adding them to the front of the list, we insert the children nodes in the list such that the updated list is maintained in ascending order of costs.

4.3. ϵ-suboptimal search

We may significantly reduce the computational effort of finding a sensor sequence if we relax the requirement of optimality. Using an ϵ-suboptimal search, it is possible to find a good sequence that does not significantly increase the scheduler cost. The cost c_{sub} obtained by an ϵ-suboptimal search always satisfies $c_{\text{sub}} < c_{\text{best}}(1+\epsilon)$, where c_{best} is the cost of the optimal sequence. In our pruning algorithm, the ϵ-suboptimal search is implemented by dividing $c_{\min}$ by $1+\epsilon$, and using the resulting value to prune the sensor sequences. This is equivalent to making the lower bound of the nodes tighter by a factor of $1+\epsilon$. We found through simulations that $0 < \epsilon < 0.2$ is an acceptable choice, and that for these values, the increase in cost over the optimal solution is approximately $35\,\epsilon\%$ (e.g., $\epsilon = 0.2$ generally gives a solution within 7% of the optimal cost).

5. SIMULATIONS AND RESULTS

We used Monte Carlo (MC) simulations to evaluate the performance of the sensor scheduling algorithms for the target/torpedo scenario described in Section 2.1. The initial target position and velocity are $(x, y) = (2000, 2500)$ m and $(\dot{x}, \dot{y}) = (-4.5, -4.5)$ m/s, respectively; the average speed of the target corresponds to 6.36 m/s (12.18 knots). The target travels for 40 time-steps of one second each, and a single bearing measurement is obtained in each time step; the standard deviation of the measurement error is 0.035 radians (2°). The torpedo and its sensor are initially located at $(2100, 2300)$ m and move $b = 15$ m in each one-second time step (a speed of 28.73 knots).

In the particle filter tracker, we used $N = 2500$ particles. The number of particles was chosen such that further

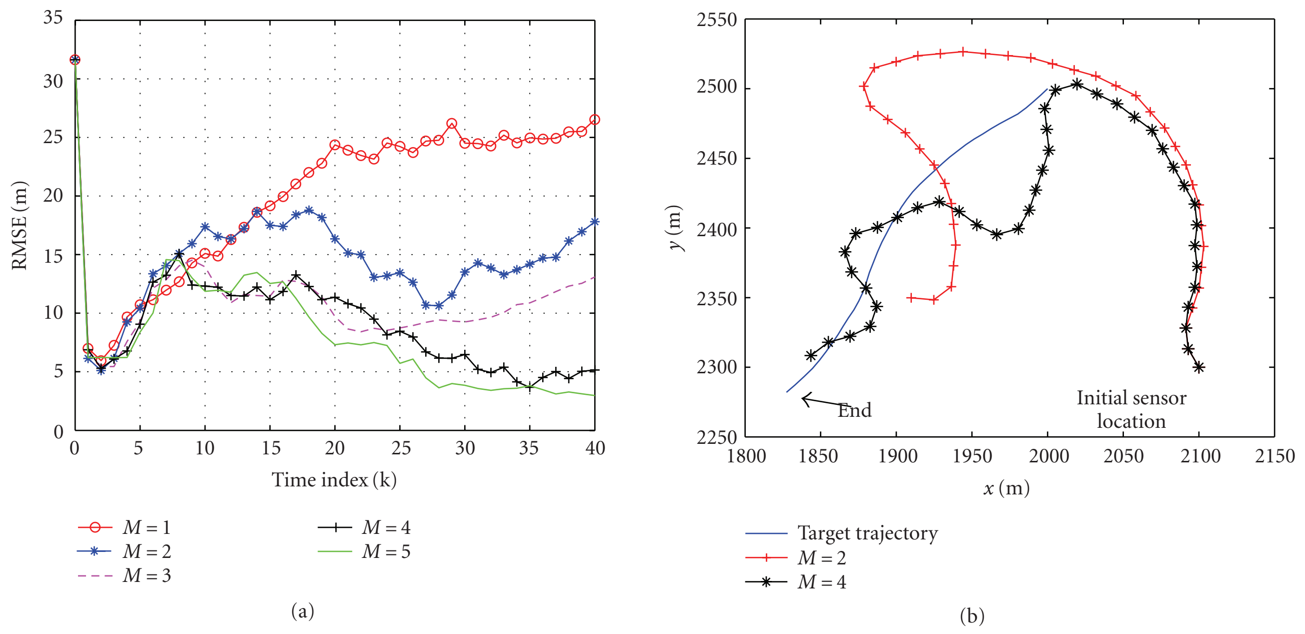


FIGURE 4: (a) Comparison of the RMSE of the target position estimate for $M = 1, 2, \ldots, 5$ OL scheduling using the UTB algorithm with the determinant cost. (b) Comparison of the sensor trajectories for the $M = 2$ and $M = 4$ OL scheduling using the UTB algorithm with the determinant cost.

increase in N does not bring a significant improvement in the tracking performance. The particles were initialized using a Gaussian density whose mean was the true target state and whose covariance was diag(500 10 500 10). The process covariance matrix in (1) was chosen as

$$\mathbf{Q} = \begin{bmatrix} 0.0667 & 0.1 & 0 & 0 \\ 0.1 & 0.2 & 0 & 0 \\ 0 & 0 & 0.0667 & 0.1 \\ 0 & 0 & 0.1 & 0.2 \end{bmatrix}. \quad (21)$$

Also, 100 MC simulations were performed for each set of parameter values.

5.1. Nonmyopic scheduling results

5.1.1. Open-loop scheduling

In this simulation, we investigated the performance of OL scheduling for values of M from 1 to 5; nonmyopic scheduling provided improved performance. We used the UTB algorithm with the determinant cost to obtain the sensor configuration sequence. The UTB parameters were chosen as $L = 2000$, $a_0 = 0.1$, $\kappa = 0$, and $\eta = 20$ (refer to Section 3.2.2).

Figure 4(a) compares root mean-square error (RMSE) of the target position for $M = 1, \ldots, 5$. It can be seen that as M increases, the RMSE performance improves, and it begins to saturate with increasing M. The RMSE curve for $M = 4$ step scheduling is on an average 2 m higher than that for the $M = 5$ step scheduling, but has a much lower computational cost; we can conclude that for the current tracking scenario, $M = 4$ step scheduling may suffice.

Figure 4(b) compares the sensor trajectory of one of the MC runs for $M = 2$ and $M = 4$ step scheduling. Initially, the sensor trajectory is similar; both schedulers use the same trajectory to reduce the initial high uncertainty about the target position. After about 16 s, the trajectories begin to differ. When $M = 4$, the sensor remains in the vicinity of the target; however, when $M = 2$, the sensor cannot plan far enough ahead to maintain a close proximity to the target. This is due to the constrained sensor movement, which leads to poor RMSE tracking performance in comparison to the case with $M = 4$.

We also performed scheduling with the KL distance cost for $M = 1, \ldots, 3$ using the UTB algorithm. Figure 5 compares the RMSE results (marked as KL $M = 1, \ldots, 3$); the RMSE performance improves with increasing M. For comparisons, we also include the RMSE result obtained with the determinant cost with $M = 3$ (labeled as Det $M = 3$). The KL scheduling resulted in slightly better RMSE performance than the determinant cost. This is possibly because the KL scheduling uses the complete statistics of the predicted state densities, while the determinant scheduling uses only up to second-order statistics (through the predicted covariance matrix).

5.1.2. Open-loop feedback scheduling

We also investigated the performance for scheduling with OLF (refer to Section 3). We used the UTB algorithm with the determinant cost. Figure 6(a) compares the RMSE results for OLF scheduling for $M = 2, 3$, and 4. It can be seen here that the RMSE performance improves as M increases. Figures 6(b), 6(c), and 6(d) compare the RMSE results of the OL and

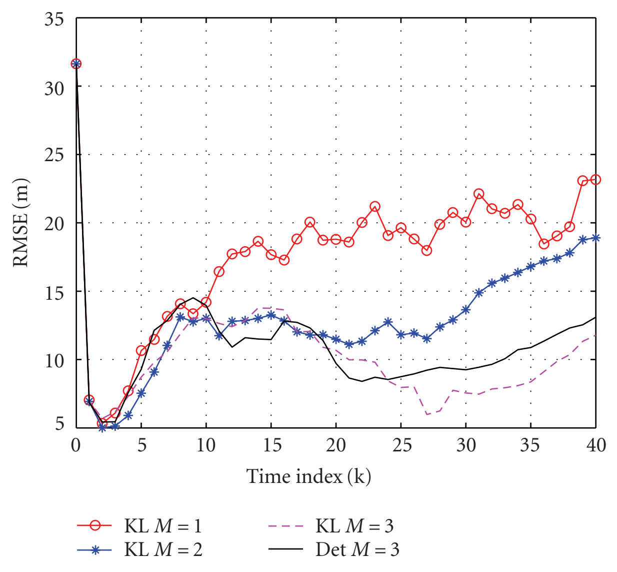


Figure 5: Comparison of the RMSE of the target position estimate for $M = 1, 2$, and 3 with the KL distance cost, using the UTB algorithm. The $M = 3$ case with the determinant cost is included for comparison.

OLF scheduling for $M = 2, 3$, and 4. It can be seen that the OLF scheduling performs better in all the cases. This is because OLF improves its scheduling decisions using the feedback provided by the measurement at each time-step. This however results in a higher computational cost (than the OL scheduling) as M-step scheduling is performed at each time-step.

5.2. Comparisons of UTB and CB scheduling

Next, we compare the UTB and CB OL scheduling results for the tracking example with the determinant cost. Figure 7 compares the RMSE performance for the CB and UTB algorithms for $M = 3$ and $M = 4$. It can be seen that the UTB algorithm yields slightly better RMSE performance than the CB algorithm. For example, when $M = 4$, the RMSE curve for the UTB algorithm is on an average 2 m lower than the RMSE curve for the CB algorithm. However, the CB algorithm is computationally more efficient than the UTB algorithm. For instance, on a Pentium IV 2.4 GHz computer with Matlab software, an exhaustive search for $M = 3$ step scheduling requires 236 s with the UTB algorithm, but only 9 s with the CB algorithm. Further, the CB algorithm requires 56 bytes for each node to store a mean and covariance while the UTB algorithm requires 1.52 KB for each node to store the measurement set C. The reduced processing-time and memory requirements make the CB algorithm a better choice for computationally constrained tracking systems.

5.3. Pruning results

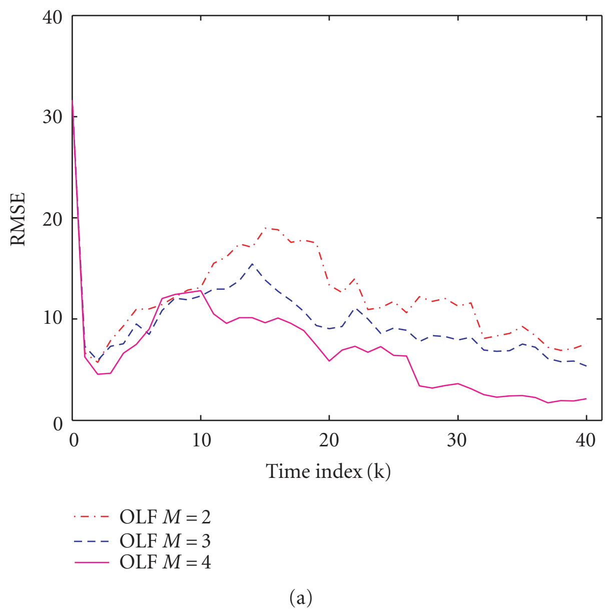
We conducted three sets of Monte Carlo experiments to evaluate the effectiveness of pruning in reducing the scheduling computational load. The first set of experiments investigated whether the optimal sensor sequence performs significantly better in terms of scheduler cost than sequences found using a suboptimal heuristic search technique; we found the optimal sequence to be significantly better. The second set of experiments investigated the relative computational requirements of the BFS-GS and UCS B&B algorithms. The third set of experiments investigated the tracking performance/computation tradeoffs of the ϵ-suboptimal search.

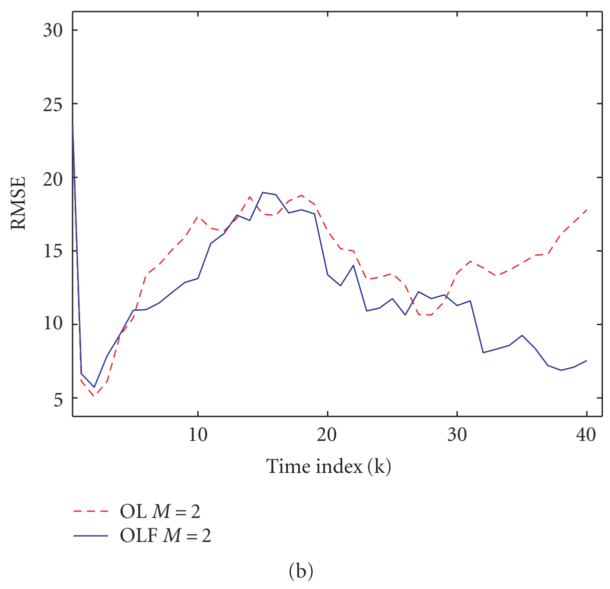
5.3.1. *Comparison of suboptimal heuristic and optimal algorithms*

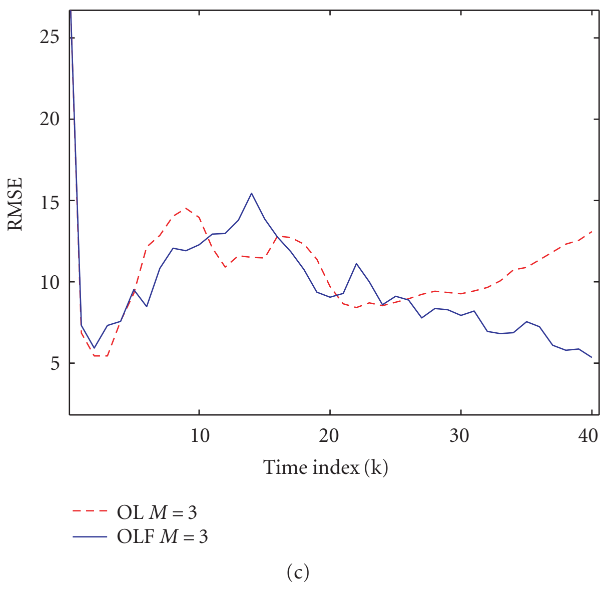
In order to assess the tracking performance of suboptimal search, we first compare the tracking performance of $M = 4$ OL optimal scheduling with the tracking performance of $M = 4$ OL suboptimal scheduling using the following heuristic: the first candidate solution obtained with a BFS up to depth d_1 is the starting point for a pure GS from depth d_1 to depth $M = 4$. We use the abbreviation Sub$[d_1, M]$ to denote this search. Figure 8(a) compares the results obtained with Sub$[1, 4]$, Sub$[2, 4]$, and Sub$[3, 4]$, and the optimal $M = 4$ scheduling. We note that the optimal $M = 4$ scheduling performs best, followed by Sub$[3, 4]$, Sub$[2, 4]$, and Sub$[1, 4]$. The relatively poor tracking performance of the suboptimal heuristic search motivated the development of the B&B pruning algorithms.

5.3.2. *Comparison of BFS-GS and UCS pruning algorithms*

We now compare the computational resources required for the two pruning algorithms described in Section 4 with the resources needed for exhaustive search. We compare the two pruning algorithms on the basis of the number of nodes opened and the maximum number of nodes stored for one M-step search averaged over all MC iterations. Table 2 summarizes the search statistics for $M = 2, \ldots, 5$ OL scheduling with $d_{\text{int}} = \lceil M/2 \rceil$. Similar results were obtained with the CB algorithm. The maximum memory consumption for the search techniques in Table 2 can be obtained by multiplying the maximum number of nodes stored with the memory usage per node. Similarly, the average scheduler time is proportional to the average number of nodes opened during the search. It can be seen that while the number of nodes opened is relatively lower for the UCS algorithm (i.e., UCS is relatively faster), the maximum number of nodes stored is relatively lower for the BFS-GS algorithm (i.e., BFS-GS is relatively less demanding in memory). This memory and scheduler-time tradeoff can be used to select between the pruning algorithms based on the tracking resources. Both algorithms require significantly less memory and less time than the exhaustive-search. We also note that as M increases, the percentage of nodes opened with both the pruning algorithms decreases. This is because with increasing M ($M \geq 4$ here), the best sensor sequence at depths close to M begins to dominate the outcome of the best sensor sequence of length M. This is consistent with Figure 4(a) where we observe that the RMSE performance begins to saturate with increasing M. Lastly, as the pruning algorithms are optimal, their RMSE







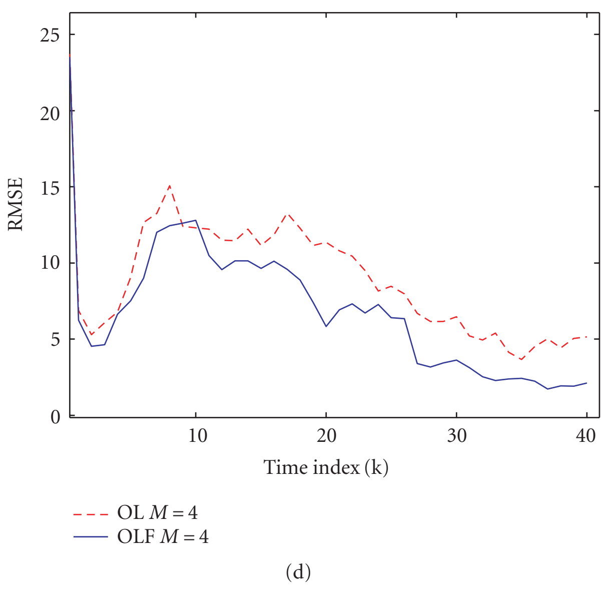


FIGURE 6: (a) RMSE comparison for OLF scheduling with $M = 2, 3$, and 4, using the UTB algorithm and the determinant cost. RMSE comparison of OLF and OL scheduling for (b) $M = 2$, (c) $M = 3$, (d) $M = 4$.

tracking performances are the same as those presented in Section 5.1.1.

5.3.3. ϵ-suboptimal algorithms

Here, we present results obtained with the ϵ-suboptimal search in Section 4.2. We performed $M = 4$ OL scheduling for different values of ϵ using the UTB algorithm and the two pruning algorithms. Figure 8(b) depicts the RMSE curves obtained with the UCS algorithm while Figures 9(a) and 9(b) depict the resource savings for different values of ϵ with the BFS-GS and UCS algorithms, respectively. Clearly, with increasing ϵ, the RMSE performance degrades, as expected. However, as seen in Figures 9(a) and 9(b), the computational savings increase with increasing ϵ. We note that a good compromise between RMSE performance and computational savings can be achieved by using $\epsilon = 0.2$.

6. DISCUSSIONS AND CONCLUSIONS

Our objective in this paper was to significantly improve the RMSE tracking performance of a constrained tracking scenario using nonmyopic scheduling. We demonstrated the improved performance using Monte Carlo simulations for the two new scheduling algorithms: the covariance-based (CB) and the unscented transform-based (UTB) algorithms.

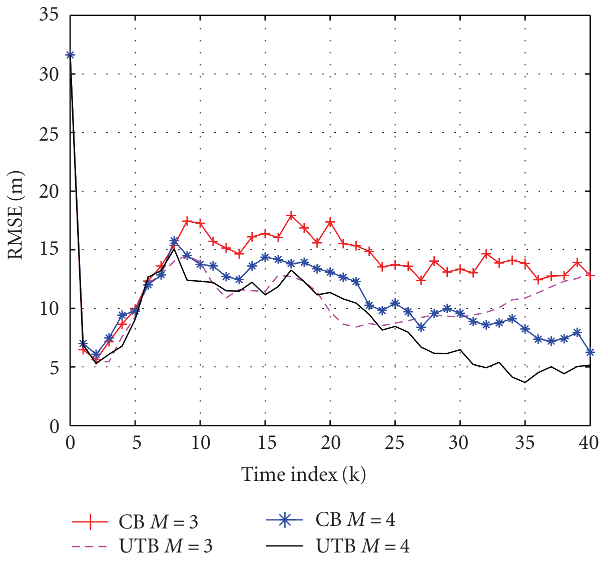


Figure 7: Comparison of the RMSE of the target position estimate using the UTB and CB algorithms for $M = 3$ and 4 with the determinant cost.

We compared two nonmyopic scheduling schemes in this paper: open-loop (OL) scheduling and open-loop feedback (OLF) scheduling. We demonstrated that while the RMSE performance of the OLF scheduling is better than the OL scheduling, the OL is computationally less intensive than the OLF. Thus, we can choose which scheduling to use based on the available computational resources. We also noted that while the UTB algorithm performs slightly better than the CB algorithm and can use arbitrary costs, the CB algorithm is computationally much more efficient, and is thus more desirable for computationally constrained tracking systems.

To further reduce the computational cost of nonmyopic scheduling, we also proposed two branch-and-bound (B&B) based optimal pruning algorithms. These algorithms are optimal in the sense that they provide the same best sensor sequence as the one obtained using an exhaustive search. We implemented the proposed pruning algorithms for a bearing-only tracking scenario and demonstrated their advantage over exhaustive search in terms of significant savings in memory and scheduling time. Our simulation results also showed that while the BFS-GS pruning algorithm is relatively memory efficient, the UCS pruning algorithm is relatively efficient in scheduler time.

Note that in future work, we plan to increase the dimensionality of the problem by increasing the number of sensors and sensor configurations. As this would significantly increase the computational requirements for optimization, we will investigate efficient search algorithms for sensor scheduling that do not require a complete enumeration of the search space. This is motivated by some of the recent developments in Q-value function approximation methods for rollout algorithms used in stochastic scheduling and stochastic shortest-path problems [18, 36]. The use of approximation techniques for scheduling in this case could be useful as the increased dimensionality can add redundancy into the information gathered from the different sensing options.

APPENDICES

A. DERIVATION OF CONDITIONAL KL DISTANCE LOST

We derive the conditional KL distance cost in (18). The KL distance conditioned on $\mathcal{Z}^j_{k+m}$ and s_{k+m} is

$$\begin{aligned}\mathcal{C}_{\mathrm{KL}}\left(\mathcal{Z}^j_{k+m}, s_{k+m}\right) &= -\int \hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m}, S_{k+m}\right)\\ &\quad\times \log\left[\frac{\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m}, \mathcal{S}_{k+m}\right)}{\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right)}\right] d\mathbf{x}_{k+m}.\end{aligned} \tag{A.1}$$

The argument of the logarithm can be further simplified using the first-order Markovian property of the dynamics model:

$$\begin{aligned}&\frac{\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m}, \mathcal{S}_{k+m}\right)}{\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right)}\\ &\quad= \frac{\hat{p}\left(z^j_{k+m} \mid \mathbf{x}_{k+m}, s_{k+m}\right)\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right)}{\hat{p}\left(z^j_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m}\right)\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right)}\\ &\quad= \frac{\hat{p}\left(z^j_{k+m} \mid \mathbf{x}_{k+m}, s_{k+m}\right)}{\int \hat{p}\left(z^j_{k+m} \mid \mathbf{x}_{k+m}, s_{k+m}\right)\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right)d\mathbf{x}_{k+m}}.\end{aligned} \tag{A.2}$$

The particles of the sets D_{k+m} along with the weights $w^{j,l}_{k+m-1}$ approximate $\hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1})$ as

$$\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right) \approx \sum_{l=1}^{L} w^{j,l}_{k+m}\delta\left(\mathbf{x}_{k+m} - \mathbf{x}^l_{k+m}\right). \tag{A.3}$$

Using (A.3), the integral in the denominator of (A.2) can be approximated as

$$\begin{aligned}&\int \hat{p}\left(z^j_{k+m} \mid \mathbf{x}_{k+m}, s_{k+m}\right)\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m-1}, \mathcal{S}_{k+m-1}\right)d\mathbf{x}_{k+m}\\ &\quad\approx \sum_{l=1}^{L} w^{j,l}_{k+m}\hat{p}\left(z^j_{k+m} \mid \mathbf{x}^l_{k+m}, s_{k+m}\right).\end{aligned} \tag{A.4}$$

The particles of set D_{k+m} along with weights $w^{j,l}_{k+m}$ approximate $\hat{p}(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m}, \mathcal{S}_{k+m})$

$$\hat{p}\left(\mathbf{x}_{k+m} \mid \mathcal{Z}^j_{k+m}, \mathcal{S}_{k+m}\right) = \sum_{l=1}^{L} w^{j,l}_{k+m}\delta\left(\mathbf{x}_{k+m} - \mathbf{x}^l_{k+m}\right). \tag{A.5}$$

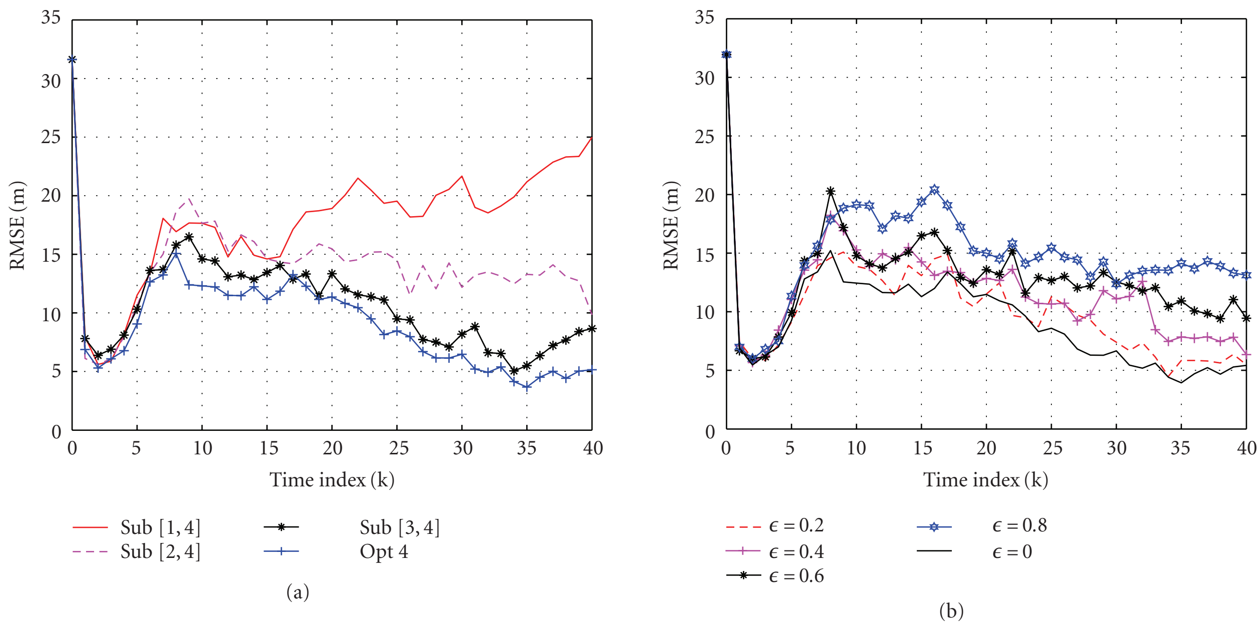


FIGURE 8: (a) Comparison of the suboptimal heuristic and optimal searches with the UTB algorithm and the determinant cost for $M = 4$ OL scheduling. Here, Sub$[d_1, 4]$ is a heuristic suboptimal algorithm composed of a BFS up to depth d_1 and pure GS from depth d_1 to depth $M = 4$. (b) Comparison of RMSE of the target position estimate for $M = 4$ and varying values of suboptimal parameter ϵ using the UCS pruning algorithm.

TABLE 2: Statistics for two pruning algorithms: BFS-GS and UCS.

M	Statistics	BFS-GS	UCS	Exhaustive search
$M = 2$	Nodes opened	81 (90%)	81 (90%)	90
	Maximum nodes stored	9	9	9
$M = 3$	Nodes opened	376 (44.75%)	313 (38.17%)	820
	Maximum nodes stored	81	71	81
$M = 4$	Nodes opened	1667 (22.59%)	1102 (14.93%)	7381
	Maximum nodes stored	89	213	729
$M = 5$	Nodes opened	3764 (5.66%)	2548 (3.84%)	66430
	Maximum nodes stored	737	1206	6561

Now substituting (A.2), (A.4), and (A.5) in (A.1), we obtain

$$\begin{aligned}
&\mathcal{C}_{\mathrm{KL}}\left(\mathcal{Z}^{j}_{k+m}, s_{k+m}\right)\\
&\quad = -\sum_{l=1}^{L} w^{j,l}_{k+m} \log \frac{\hat{p}\left(z^{j}_{k+m} \mid \mathbf{x}^{l}_{k+m}, s_{k+m}\right)}{\sum_{l=1}^{L} w^{j,l}_{k+m-1}\hat{p}\left(z^{j}_{k+m} \mid \mathbf{x}^{l}_{k+m}, s_{k+m}\right)}\\
&\quad = -\sum_{l=1}^{L} w^{j,l}_{k+m} \log \frac{\hat{p}\left(z^{j}_{k+m} \mid \mathbf{x}^{l}_{k+m}, s_{k+m}\right) w^{j,l}_{k+m-1}}{\left(\sum_{l=1}^{L} w^{j,l}_{k+m-1}\hat{p}\left(z^{j}_{k+m} \mid \mathbf{x}^{l}_{k+m}, s_{k+m}\right)\right) w^{j,l}_{k+m-1}}\\
&\quad = -\sum_{l=1}^{L} w^{j,l}_{k+m} \log \frac{w^{j,l}_{k+m}}{w^{j,l}_{k+m-1}}.
\end{aligned} \tag{A.6}$$

B. DERIVATION OF RECURSIVE WEIGHT UPDATE EQUATION

We derive the weight update step of equation (i) of Algorithm 2. For this, we first define an augmented state particle $\mathcal{X}^{l}_{k+m} \triangleq [\mathbf{x}^{l}_{k+1} \cdots \mathbf{x}^{l}_{k+m}]^{T}$ as the vector obtained by concatenating the predicted state particles $\mathbf{x}^{l}_{k+r}$, $r = 1, \ldots, m$. The augmented state particle is obtained by sampling from the kinematic prior distribution $p(\mathbf{x}_{k+m} \mid \mathbf{x}^{l}_{k+m-1}) \cdots p(\mathbf{x}_{k+1} \mid \mathbf{x}^{l}_{k}) \cdot p(\mathbf{x}^{l}_{k} \mid Z_k, S_k)$. Similarly, $\mathcal{Z}^{j}_{k+m} = [z^{j}_{k+1} \cdots z^{j}_{k+m}]^{T}$ is an augmented measurement particle that is obtained by concatenating the predicted measurement samples z^{j}_{k+r}, $r = 1, \ldots, m$.

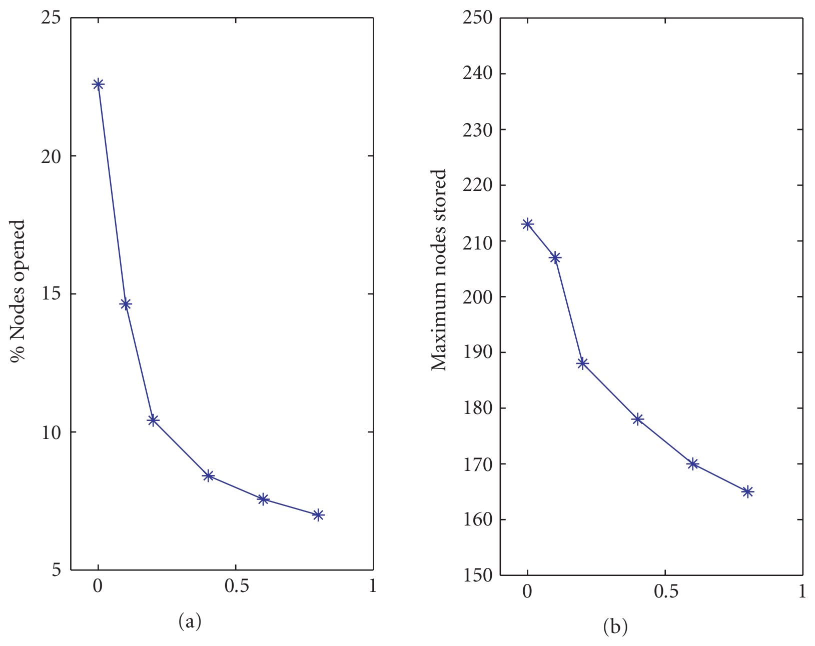


FIGURE 9: (a) Percentage of nodes opened as a function of ϵ for $M = 4$ with the BFS-GS pruning algorithm. (b) Maximum number of nodes stored as a function of ϵ for $M = 4$ with the UCS pruning algorithm.

For a given $\mathcal{Z}^j_{k+m}$ we can obtain the following discrete approximation of the posterior density:

$$p\left(\mathcal{X}_{k+m} \mid \mathcal{Z}^j_{k+m}, \mathcal{S}_{k+m}, Z_k, S_k\right) \approx \sum_{l=1}^{L} w^{j,l}_{k+m}\delta\left(\mathcal{X}_{k+m} - \mathcal{X}^l_{k+m}\right), \tag{B.1}$$

where we have from [17]

$$\begin{aligned} w^{j,l}_{k+m} &\propto p\left(\mathcal{Z}^j_{k+m} \mid \mathcal{X}^l_{k+m}, \mathcal{S}_{k+m}\right) \\ &= p\left(z^j_{k+m} \mid \mathcal{X}^l_{k+m}, \mathcal{S}_{k+m}, \mathcal{Z}^j_{k+m-1}\right) p\left(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m}, \mathcal{S}_{k+m}\right) \\ &= p\left(z^j_{k+m} \mid \mathbf{x}^l_{k+m}, s_{k+m}\right) p\left(\mathcal{Z}^j_{k+m-1} \mid \mathcal{X}^l_{k+m-1}, \mathcal{S}_{k+m-1}\right). \end{aligned} \tag{B.2}$$

ACKNOWLEDGMENT

This work was supported by the DARPA Integrated Sensing and Processing Program through a contract with Office of Naval Research N00014-04-0437.

REFERENCES

[1] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1382–1397, 2002.

[2] D. A. Castanon, "Approximate dynamic programming for sensor management," in *Proceedings of the 36th IEEE International Conference on Decision and Control*, vol. 2, pp. 1202–1207, San Diego, Calif, USA, December 1997.

[3] A. Doucet, B.-N. Vo, C. Andrieu, and M. Davy, "Particle filtering for multi-target tracking and sensor management," in *Proceedings of the 5th International Conference on Information Fusion*, vol. 1, pp. 474–481, Annapolis, Md, USA, July 2002.

[4] C. Kreucher, K. Kastella, and A. O. Hero III, "A Bayesian method for integrated multitarget tracking and sensor management," in *Proceedings of the 6th International Conference on Information Fusion*, pp. 132–139, Cairns, Qld., Australia, July 2003.

[5] A. Logothetis and A. Isaksson, "On sensor scheduling via information theoretic criteria," in *Proceedings of the American Control Conference*, vol. 4, pp. 2402–2406, San Diego, Calif, USA, June 1999.

[6] S. Singh, B.-N. Vo, A. Doucet, and R. Evans, "Stochastic approximation for optimal observer trajectory planning," in *Proceedings of 42nd IEEE International Conference on Decision and Control*, vol. 6, pp. 6313–6318, Maui, Hawaii, USA, December 2003.

[7] M. Kalandros and L. Y. Pao, "Covariance control for multisensor systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 4, pp. 1138–1157, 2002.

[8] M. L. Hernandez, T. Kirubarajan, and Y. Bar-Shalom, "Multisensor resource deployment using posterior Cramér-Rao bounds," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, pp. 399–416, 2004.

[9] M. L. Hernandez, "Optimal sensor trajectories in bearings-only tracking," in *Proceedings of the 7th International Conference on Information Fusion*, vol. 2, pp. 893–900, Stockholm, Sweden, June-July 2004.

[10] J. Liu, D. Petrovic, and F. Zhao, "Multi-step information-directed sensor querying in distributed sensor networks," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 5, pp. 145–148, Hong Kong, April 2003.

[11] R. E. Korf, "Artificial intelligence search algorithms," in *CRC Handbook of Algorithms and Theory of Computation*, chapter 36, pp. 1–20, CRC Press, Boca Raton, Fla, USA, 1998.

[12] N. R. Vempaty, V. Kumar, and R. E. Korf, "Depth-first vs. best-first search," in *Proceedings of 9th National Conference on Artificial Intelligence (AAAI '91)*, vol. 1, pp. 434–440, Anaheim, Calif, USA, July 1991.

[13] R. Battiti and G. Tecchiolli, "The reactive Tabu search," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 126–140, 1994.

[14] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray, "Sensor scheduling algorithms requiring limited computation [vehicle sonar range-finder example]," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 3, pp. 825–828, Montreal, Quebec, Canada, May 2004.

[15] R. D. Short, "Sting Ray—a sound-seeking missile," *IEE Review*, vol. 35, no. 11, pp. 419–423, 1989.

[16] A. S. Chhetri, D. Morrell, C. Chakrabarti, A. Papandreou-Suppappola, A. Spanias, and J. Zhang, "A unified Bayesian decision theory perspective to sensor networks," in *Proceedings of the Joint International Symposium on Intelligent Control & 13th Mediterranean Conference on Control and Automation (ISIC-MED '05)*, pp. 598–603, Limassol, Cyprus, June 2005.

[17] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[18] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1 & 2, Athena Scientific, Belmont, Mass, USA, 2001.

[19] R. Ku and M. Athans, "On the adaptive control of linear systems using the open-loop-feedback-optimal approach," *IEEE Transactions on Automatic Control*, vol. 18, no. 5, pp. 489–493, 1973.

[20] E. Tse and M. Athans, "Adaptive stochastic control for a class of linear systems," *IEEE Transactions on Automatic Control*, vol. 17, no. 1, pp. 38–52, 1972.

[21] F. Balduzzi and G. Menga, "Open-loop feedback control for hybrid manufacturing systems," in *Proceedings of the 39th IEEE International Conference on Decision and Control*, vol. 4, pp. 3144–3150, Sydney, NSW, Australia, December 2000.

[22] E. Mosca, *Optimal, Predictive, and Adaptive Control*, Prentice-Hall, Upper Saddle River, NJ, USA, 1994.

[23] Y. Nakamura, "Geometric fusion: minimizing uncertainty volumes," in *Data Fusion in Robotics and Machine Intelligence*, pp. 457–480, Academic Press, Boston, Mass, USA, 1992.

[24] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: an information-directed approach," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1199–1209, 2003.

[25] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 36–45, Berkeley, Calif, USA, April 2004.

[26] J. S. Manyika and H. F. Durrant-Whyte, *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*, Ellis Horwood, New York, NY, USA, 1994.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, NY, USA, 1991.

[28] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "Scheduling multiple sensors using particle filters in target tracking," in *Proceedings of IEEE Workshop on Statistical Signal Processing*, pp. 529–532, St. Louis, Mo, USA, September 2003.

[29] J. R. J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 4, pp. 378–391, 2003.

[30] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "The use of particle filtering with the unscented transform to schedule sensors multiple steps ahead," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 2, pp. 301–304, Montreal, Quebec, Canada, May 2004.

[31] S. J. Julier and J. K. Uhlmann, "Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations," in *Proceedings of the American Control Conference*, vol. 2, pp. 887–892, Anchorage, Alaska, USA, May 2002.

[32] A. S. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "Efficient search strategies for non-myopic sensor scheduling in target tracking," in *Conference Record of the 38th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 2106–2110, Pacific Grove, Calif, USA, November 2004.

[33] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "On the solution of traveling salesman problems," *Documenta Mathematica. Deutsche Mathematiker-Vereinigung*, vol. 3, pp. 645–656, 1998.

[34] M. L. Fisher, "Optimal solution of vehicle routing problems using minimum *k*-trees," *Operations Research*, vol. 42, no. 4, pp. 626–642, 1994.

[35] T. M. Breuel, "A comparison of search strategies for geometric branch and bound algorithms," in *Proceedings of 7th European Conference on Computer Vision (ECCV '02)*, vol. 3, pp. 837–850, Copenhagen, Denmark, May 2002.

[36] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.

Amit S. Chhetri received his B.Tech degree from the Department of Electrical Engineering, Indian Institute of Technology, Kanpur, in May 2000, and his M.S.E.E. degree from the Department of Electrical and Computer Engineering, University of Rhode Island, in May 2002. He is currently pursuing his Ph.D. degree in the Department of Electrical Engineering at Arizona State University. From May 2004 to August 2004, and from May 2005 to August 2005, he worked as a Research Fellow with the Knowledge Tools Group at Microsoft Research, Redmond. His research interests include integrated sensing and processing and target tracking.

Darryl Morrell received his B.S, M.S, and Ph.D. degrees in electrical engineering from Brigham Young University in 1984, 1986, and 1988. He joined the Department of Electrical Engineering at Arizona State University in 1988; in 2004, he joined the Department of Engineering at Arizona State University at the Polytechnic Campus, where he is currently an Associate Professor. His research interests include statistical decision and estimation theory and stochastic filtering, with application to sensing and target tracking problems.

Antonia Papandreou-Suppappola received her Ph.D. degree in electrical engineering in 1995 from the University of Rhode Island before holding a navy-supported research faculty position. She is currently an Associate Professor at Arizona State University. Her research interests are in the areas of time-frequency signal processing, integrated sensing and processing, and signal processing for wireless communications. Her publication record consists of more than eighty refereed journal articles, book chapters, and conference papers. She is the recipient of the NSF Career Award in 2002, and she is currently serving as an Associate Editor for the IEEE Transactions on Signal Processing and as the Treasurer of the Conference Board of the IEEE Signal Processing Society.