Received May 24, 2013, accepted June 11, 2013, date of publication July 10, 2013, date of current version July 17, 2013.

Digital Object Identifier 10.1109/ACCESS.2013.2272664

Adaptive Estimation of Time-Varying Sparse Signals

RAMIN ZAHEDI (Member, IEEE)[1], LUCAS W. KRAKOW (Member, IEEE)[1], EDWIN K. P. CHONG (Fellow, IEEE)[1,2], AND ALI PEZESHKI (Member, IEEE)[1,2]

[1]Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA
[2]Department of Mathematics, Colorado State University, Fort Collins, CO 80523, USA

Corresponding author: R. Zahedi (ramin.zahedi@colostate.edu)

This work was supported in part by DARPA/DSO under Contract N66001-11-C-4023, ONR Contract N00014-08-1-110, and AFOSR Contract FA9550-09-1-0518, the National Science Foundation under Grants CCF-0916314 and CCF-1018472, and the Colorado State University Libraries Open Access Research and Scholarship Fund.

ABSTRACT We consider the problem of adaptively designing compressive measurement matrices for estimating time-varying sparse signals. We formulate this problem as a *partially observable Markov decision process*. This formulation allows us to use Bellman's principle of optimality in the implementation of multi-step lookahead designs of compressive measurements. We compare the performance of adaptive versus traditional non-adaptive designs and study the value of multi-step (non-myopic) versus one-step (myopic) lookahead adaptive schemes by introducing two variations of the compressive measurement design problem. In the first variation, we consider the problem of sequentially selecting measurement matrices with fixed dimensions from a prespecified library of measurement matrices. In the second variation, the number of compressive measurements, i.e., the number of rows of the measurement matrix, is adaptively chosen. Once the number of measurements is determined, the matrix entries are chosen according to a prespecified adaptive scheme. Each of these two problems is judged by a separate performance criterion. The gauge of efficiency in the first problem is the conditional mutual information between the sparse signal support and measurements. The second problem applies a linear combination of the number of measurements and conditional mutual information as the performance measure. Through several simulations, we study the effectiveness of different designs in various settings. The primary focus in these simulations is the application of a method known as *rollout*. However, the computational load required for using the rollout method has also inspired us to adapt two data association heuristics to the compressive sensing paradigm. These heuristics show promising decreases in the amount of computation for propagating distributions and searching for optimal solutions.

INDEX TERMS Adaptive compressive sensing, data association, multi-target tracking, POMDP, Q-value approximation, rollout.

I. INTRODUCTION

Compressive Sensing (CS) concentrates on solving the problem of recovering sparse (or compressible) signals using linear compressive measurements (see, e.g., [1]–[5]). CS results suggest that a sparse signal can be completely recovered if a sufficient number of measurements, but much lower than the sparse signal dimension, are made in a noiseless environment. In particular, these results show that the exact recovery of the sparse signal can be achieved with a high probability if compressive measurement matrices with random Gaussian or Bernoulli entries are used for measuring the signal.

Since then, numerous methods for designing the compressive measurement matrix have been proposed (see, e.g., [6]–[11]). For example, the work in [10] provides a deterministic structure for the measurement matrix which requires more measurements to recover the sparse signal but the overall computational cost is much lower than the random design. Recently, more realistic settings for the sparse signal recovery problem have been considered (see, e.g., [12]–[14]), opening new horizons for alternative compressive measurement designs.

The traditional CS methods are non-adaptive in the sense that the measurement matrix is determined in advance, prior to receiving any measurements of the sparse signal. The recent theoretical result in [15] shows that, in a certain asymptotic sense, adaptive designs have limited performance advantage over traditional non-adaptive schemes. On the other hand, several works (see, e.g., [16]–[20]) have proposed adap-

tive designs for certain scenarios, showing (non-asymptotic) improvements resulting from adaptation under these scenarios.

The above works all assume a *static* sparse signal. This means that over time, the locations and values of the nonzero entries of the sparse signal stay constant. Although this assumption is valid in some applications, in many practical scenarios, considering a static signal seems unrealistic. Examples of time-varying sparse signals arise naturally in target tracking, high-speed video capturing, time-varying multipath in communication, etc. However, little attention has so far been paid to the time-varying case (notable exceptions being [21]–[29]), where one would expect adaptation to play a more significant role.

In this paper, we study the problem of adaptively designing compressive measurement matrices for estimating time-varying sparse signals. We take a unique perspective that enables melding CS and data association techniques for multi-target tracking: We view the time-varying sparse signal as a representation of the combination of target locations (the vector support) and target values (the non-zero values located at the support indices) of s targets moving over N possible locations according to a discrete motion-model.

At each time step k, the locations and strength values of these targets can be represented by an s-sparse vector $\mathbf{x}_k$ in $\mathbb{R}^N$. We call an N-dimensional signal *s-sparse* if it has at most $s \ll N$ nonzero entries. Moreover, we assume that only the target locations, or equivalently the *support* of the sparse signal $\mathbf{x}_k$, change over time and not the target values. Our goal is to estimate the support of the sparse signal at each time step from compressive measurements.

During each time step k, we collect $1 \leq l_k \leq l_{\max}$ measurements ($l_{\max}$ is a prespecified integer), represented by the l_k-vector $\mathbf{y}_k = [y_1, y_2, \ldots, y_{l_k}]^T$, according to the linear model $\mathbf{y}_k = \mathbf{A}_k\mathbf{x}_k + \mathbf{w}_k$. In this model, $\mathbf{A}_k \in \mathbb{R}^{l_k \times N}$ is a compressive measurement matrix with rows $\mathbf{A}_k(i)$, $i = 1, \ldots, l_k$, and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}_{l_k}, \sigma_w^2\mathbf{I}_{l_k})$, where $\mathcal{N}(\mathbf{0}_{l_k}, \sigma_w^2\mathbf{I}_{l_k})$ denotes the l_k-variate normal distribution with the zero mean vector $\mathbf{0}_{l_k}$ and the covariance matrix $\sigma_w^2\mathbf{I}_{l_k}$, and $\mathbf{I}_{l_k}$ is the $(l_k \times l_k)$ identity matrix. We assume that $\|\mathbf{A}_k(i)\|_2 = 1$ for $i = 1, \ldots, l_k$ and $k = 1, \ldots, m$. Also, we assume that at each time step, the movement of the targets is independent of the measurement matrix $\mathbf{A}_k$ or the number of measurements l_k. Thus, at any time step, multiple targets can move to the same location, effectively reducing the number of nonzero entries in the signal to less than s.

We now introduce two problems of interest. These problems both contain the essence of compressive sensing as described above. At the same time, each problem presents unique goals and constraints. The differences in these problems ultimately define the type of their favored solution methods.

Problem 1. Fix $l_k = l$ for $k = 1, 2, \ldots, m$, where $1 \leq l \leq l_{\max}$ is a fixed integer. Our goal is to sequentially select measurement matrices $\mathbf{A}_k$, $k = 1, 2, \ldots, m$, from a prespecified library to maximize a measure of performance for estimating locations of the nonzero entries (support) of the sparse signal $\mathbf{x}_k$ at each time step k. The performance measure for this problem is the conditional mutual information between the measurement vector $\mathbf{y}_k$ and the support of the signal $\mathbf{x}_k$ given the previous measurements.

Problem 2. Here, l_k is not fixed and is, in fact, the action input chosen sequentially at each time step, $k = 1, 2, \ldots, m$. Once the value l_k is chosen, the measurement matrix $\mathbf{A}_k$ is constructed using a prespecified scheme, explained in Section IV. The performance criterion in this problem is defined as a weighted sum of the number of measurements l_k and the conditional mutual information between the measurements and the support of the sparse signal at each time step.

The problems introduced above present two variations of support identification for sparse signals. Also, both of these problems lend themselves to adaptive measurement solutions, successively selecting action inputs as observations are collected. We formulate each problem as a finite-horizon partially observable Markov decision process (POMDP) (see [30] and [31]). This formulation enables us to bring to bear Bellman's principle of optimality. Furthermore, the POMDP framework accommodates two categories of adaptive solution methods, allowing their implementation and comparison.

When variations of Problem 1 have been considered in the CS literature, the common solution approach is categorized as a ''1-step lookahead'' or ''myopic'' method. In other words, only the performance at the current time step is considered when selecting the compressive measurement matrices. Although Section IV *includes* results from 1-step lookahead methods, our primary focus in this paper is the application of ''multi-step lookahead'' solutions.

It is important to realize that multi-step lookahead methods can perform better than myopic solutions only when there is a benefit in accounting for long-term effects. Several factors play a role in providing such benefits. The performance criterion of the problem and the action space restrictions are examples of such factors. In this paper, throughout several simulations, we show that although adaptive designs outperform the non-adaptive schemes, the performance criterion and the action space of Problem 1 preclude the multi-step lookahead solution from surpassing the myopic solutions. The alterations in Problem 2, namely the action space and the performance measure, exemplify long-term considerations where the performance of multi-step lookahead solutions exceeds that of myopic schemes. We verify this claim throughout several simulations. In fact, we have previously verified this claim for static and time-varying 1-sparse signals in noisy environments in [32] and [33].

This paper is an extension of the previous work in which we concentrate on estimating time-varying s-sparse signals with more than one nonzero entry. Note that once we assume more than one nonzero moving entry in the signal, we must consider the problem of data association (see, e.g., [34]–[36]) that naturally arises in our problem. Therefore, we have to

modify our POMDP formulation (Section II) accordingly to take this problem into account.

It is well known that solving a POMDP problem is typically computationally prohibitive (see, e.g., [37] and [38]). The following approximation techniques decrease the computation volume, alleviating some of the conventional concerns.

1) We introduce two heuristics that are motivated from the well known techniques *joint probabilistic data association* (JPDA) and *multiple hypothesis tracking* (MHT) in the target tracking literature (see, e.g., [39]–[42]) to simplify the update of the belief state in the POMDP.
2) In the multi-step lookahead variation of our method, we use an approximation method, known as *rollout* (see [43]), to estimate an optimal solution for the POMDP.

II. POMDP FORMULATION

First, we formulate the problems introduced above as POMDPs. Since most of the POMDP elements for these two problems are similar, we present one POMDP formulation, but clarify the differences in the formulation of each problem where it is necessary.

States and Transition Law: The s-sparse vector $\mathbf{x}_k$ is fully characterized by its support (the locations of the targets) and the strength values (the values of the nonzero entries). Therefore, we take the state of the POMDP at time k to be $\mathbf{s}_k = (\mathbf{d}_k, \mathbf{v}_k)$, where $\mathbf{d}_k$ and $\mathbf{v}_k$ are $(s \times 1)$ vectors. The ith entry of these vectors, $\mathbf{d}_k(i)$ and $\mathbf{v}_k(i)$, represent the location and the strength value of the ith target at time k, respectively. Let the set Ω be defined as $\Omega = \{1, \ldots, N\}$. Also, let Ω_s be the set containing all the $(s \times 1)$ vectors $\mathbf{d}$ such that $\mathbf{d}(i) \in \Omega$ for $i = 1, \ldots, s$. Note that by using this definition for Ω_s, we are allowing any group of targets to be in the same location at the same time. Therefore, the cardinality of the set Ω_s is equal to $|\Omega_s| = N^s$. The POMDP state space is then defined as the Cartesian product $\mathcal{S} = (\Omega_s \times \mathbb{R}^s)$.

The state transition law of the POMDP, defined by the movement of the targets, is independent of our actions. Although the above definition for the POMDP state allows for the strength values of the targets to change over time, we make the simplifying assumption that these values stay constant over time. Accordingly, the state transition law of the POMDP can be described as:

$$P\{\mathbf{s}_{k+1} = (\mathbf{d}, \mathbf{v}) | \mathbf{s}_k = (\mathbf{e}, \mathbf{z})\} = \begin{cases} p_{\mathbf{ed}}, & \mathbf{v} = \mathbf{z}, \\ 0, & \text{otherwise}, \end{cases}$$

where $p_{\mathbf{ed}}$ is the probability that targets in locations represented by the vector $\mathbf{e}$ transition to locations represented by the vector $\mathbf{d}$. Depending on the movement of the targets, the probabilities $p_{\mathbf{ed}}$ could have different values. For example, if we consider the case where targets stay in the same locations at all time steps, then $p_{\mathbf{ed}} = 1$ if $\mathbf{e} = \mathbf{d}$, and $p_{\mathbf{ed}} = 0$ if otherwise.

Actions: In Problem 1, the action at each time step k is the selection of the measurement matrix $\mathbf{A}_k$. Therefore, the action space $\mathcal{A}$ is a prespecified subset of all matrices $\mathbf{A} \in \mathbb{R}^{l \times N}$ such that, for each row $\mathbf{A}(i), i = 1, \ldots, l$, $\|\mathbf{A}(i)\|_2 = 1$. In Problem 2, choosing the number of measurements l_k at each time step k is the POMDP action. In this case, the action space $\mathcal{A}$ is the set of natural numbers. For simplicity, we use the notation $\mathbf{u}_k$ for the POMDP action at time k for both problems.

Observations and Observation Law: The POMDP observations are the measurements $\mathbf{y}_k$ at each time step k. The observation law can be described in the following way: Given $\mathbf{s}_k = (\mathbf{d}, \mathbf{v})$ and the matrix $\mathbf{A}_k = \mathbf{A}$ at time step k, it can be easily shown that $\mathbf{y}_k | (\mathbf{s}_k = (\mathbf{d}, \mathbf{v}), \mathbf{A}_k = \mathbf{A}) \sim \mathcal{N}(\mathbf{A}_{\mathbf{d}}\mathbf{v}, \sigma_w^2 \mathbf{I}_{l_k})$, where $\mathbf{A}_{\mathbf{d}}$ is a matrix whose ith column is the $\mathbf{d}(i)$th column of the matrix $\mathbf{A}$.

Cost: Let $I(\mathbf{y}_k; \mathbf{d}_k | H_k)$ be the conditional mutual information between the signal support $\mathbf{d}_k$ and the measurements $\mathbf{y}_k$ given the history $H_k = \{\mathbf{u}_1, \mathbf{y}_1, \ldots, \mathbf{u}_k, \mathbf{y}_k\}$. The per-time-step cost $c_k(\mathbf{s}_k, \mathbf{u}_k)$ for Problem 1 is then simply defined as $c_k(\mathbf{s}_k, \mathbf{u}_k) = -I(\mathbf{y}_k; \mathbf{d}_k | H_k)$. For Problem 2, the cost is a combination of the number of measurements l_k and the conditional mutual information $I(\mathbf{y}_k; \mathbf{d}_k | H_k)$ at each time step k. More specifically, the per-time-step cost is defined as

$$c_k(\mathbf{s}_k, \mathbf{u}_k) = l_k - \gamma I(\mathbf{y}_k; \mathbf{d}_k | H_k), \tag{1}$$

where $\gamma \geq 0$ is a weighting parameter chosen based on the priority of either l_k or $I(\mathbf{y}_k; \mathbf{d}_k | H_k)$. For example, if we are not particularly concerned with the number of measurements, then we choose a large value for γ. On the contrary, if we have to be careful with the total number of measurements used in m steps, we use a small value for γ.

Belief State: The belief state $\mathbf{b}_k$ at time k is defined as the posterior (conditional) distribution of the state $\mathbf{s}_k$ given the history H_{k-1}. Using the belief state definition, our POMDP can be presented as an equivalent Markov Decision Process (MDP) (see [44]). This MDP, which is also referred to as the *belief MDP*, has the following components:

1) A state space consisting of all the possible belief states $\mathbf{b}_k$ for the POMDP.
2) A state transition law that is computed using the state transition law and the observation law of the POMDP. This computation is referred to as the *belief state update*, shown in detail in the next section.
3) An action space which is the same as that of the POMDP.
4) A cost $C_k(\mathbf{b}_k, \mathbf{u}_k)$, referred to as the *belief cost*, which is defined as:

$$C_k(\mathbf{b}_k, \mathbf{u}_k) = \mathbf{E}\left[c_k(\mathbf{s}_k, \mathbf{u}_k) | H_k, \mathbf{b}_k\right], \tag{2}$$

where the expectation is with respect to the posterior distribution of the state $\mathbf{s}_k$ given the history H_k at time step k, i.e., the belief state $\mathbf{b}_k$.

Using the belief MDP model, we can now apply the approximation methods available in the literature, specifically rollout [45].

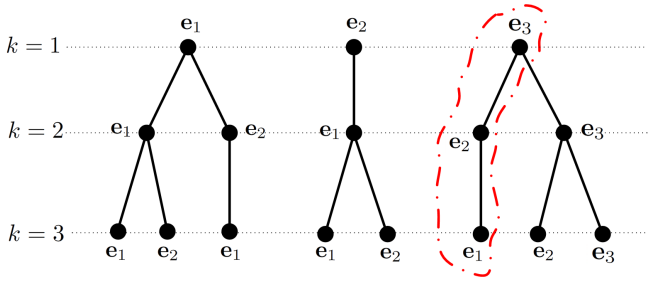


FIGURE 1. A simple example showing the construction of various links and routes from only three support possibilities $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ over time.

A. BELIEF STATE UPDATE

Let $P_{\mathbf{d}_k|H_k}$ be the conditional probability mass function of $\mathbf{d}_k$ given H_k, and $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ be the conditional density function of $\mathbf{v}_k$ given $\mathbf{d}_k$ and H_k at time step k. Consider a simple example, shown in Fig. 1, where there are three *nodes*, $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$, representing the possible signal supports at any time step. Fig. 1 also displays the *links* between nodes, representing allowed support transitions. For example, there are two possible transitions arriving at $\mathbf{e}_1$, namely, those originating from nodes $\mathbf{e}_1$ and $\mathbf{e}_2$. Attached to the initial node of each link is a weight value, computed from a prior conditional probability mass function, and also a prior conditional density function. Upon choosing an action and receiving measurements, we can compute a posterior weight value as well as a posterior density function for the terminal node of the link.

To clarify, consider the selected *route*, comprised of successive links and outlined in red in Fig. 1, that begins at node $\mathbf{e}_3$ at time $k = 1$ and after passing through node $\mathbf{e}_2$ at time $k = 2$, it finally arrives at node $\mathbf{e}_1$ at $k = 3$. At time $k = 1$, once measurement $\mathbf{y}_1$ is received, we can compute the posterior weight value $P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1)$ and the posterior density function $f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3, H_1)$ using the initial prior functions $P_{\mathbf{d}_0}$ and $f_{\mathbf{v}_0|\mathbf{d}_0}$ (this calculation is shown later in this section). Then, examining the link from $\mathbf{e}_3$ to $\mathbf{e}_2$, the prior weight and density function attached to node $\mathbf{e}_3$ for the next time step are $P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1)$ and $f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3, H_1)$, respectively. The prior values are updated using the received observations, resulting in the posterior weight and density function $P_{\mathbf{d}_1|\mathbf{d}_0,H_1}(\mathbf{e}_2|\mathbf{e}_3, H_1)$ and $f_{\mathbf{v}_1|\mathbf{d}_1,\mathbf{d}_0,H_1}(\cdot|\mathbf{e}_2, \mathbf{e}_3, H_1)$, respectively. For the next link in the route from $\mathbf{e}_2$ to $\mathbf{e}_1$, the prior weight and density function used for $\mathbf{e}_2$ will be the posterior weight and density function of this node at $k = 2$, i.e., $P_{\mathbf{d}_1|\mathbf{d}_0,H_1}(\mathbf{e}_2|\mathbf{e}_3, H_1)$ and $f_{\mathbf{v}_1|\mathbf{d}_1,\mathbf{d}_0,H_1}(\cdot|\mathbf{e}_2, \mathbf{e}_3, H_1)$, respectively. Again, after choosing an action and receiving measurements, the posterior weight $P_{\mathbf{d}_2|\mathbf{d}_1,H_2}(\mathbf{e}_1|\mathbf{e}_2, H_2)$ and posterior density function $f_{\mathbf{v}_2|\mathbf{d}_2,\mathbf{d}_1,H_2}(\cdot|\mathbf{e}_1, \mathbf{e}_2, H_2)$ for node $\mathbf{e}_1$ are computed.

It is important to realize that after the first time step, the posterior weight and density function for the ending node of any link are always conditioned on the initial node of that link. Thus, knowing the positions of the targets and their prior weight and density at the previous time step is required. At any time step k, given each possible link, we define an ordered tuple $\boldsymbol{\tau}$ consisting of four elements: the terminal node, the initial node, the prior weight value, and the prior density function attached to the initial node and let $\boldsymbol{\tau}(i)$ be the ith element of the tuple. For the same route used in the above example, the tuple defined for the link from $\mathbf{e}_3$ to $\mathbf{e}_2$ is $\boldsymbol{\tau} = \big(\mathbf{e}_2, \mathbf{e}_3, P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1), f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3, H_1)\big)$. Analogously, the tuple for the last link in this route, from $\mathbf{e}_2$ to $\mathbf{e}_1$, is $\boldsymbol{\tau} = \big(\mathbf{e}_1, \mathbf{e}_2, P_{\mathbf{d}_2|\mathbf{d}_1,H_2}(\mathbf{e}_1|\mathbf{e}_2, H_2), f_{\mathbf{v}_2|\mathbf{d}_2,\mathbf{d}_1,H_2}(\cdot|\mathbf{e}_1, \mathbf{e}_2, H_2)\big)$.

At any time step, multiple tuples with the same initial and terminal nodes may exist. For example, in Fig. 1, from time $k = 2$ to time $k = 3$, there are two links starting at $\mathbf{e}_2$ and ending at $\mathbf{e}_1$. These two links can be distinguished by the unique priors attached to the starting nodes of either link. Let $\mathcal{T}_k$ be the set containing all the possible tuples at time step k. Also, define $\mathcal{T}_{k,\mathbf{d}}$ to be $\mathcal{T}_{k,\mathbf{d}} = \{\boldsymbol{\tau} : \boldsymbol{\tau} \in \mathcal{T}_k, \boldsymbol{\tau}(1) = \mathbf{d}\}$, the set of all tuples representing links ending with support $\mathbf{d}$ at time k. At each time step k, the number of possible links, which depends on the number of initial nodes and the state transition law, determines the cardinality $|\mathcal{T}_k|$ of the set $\mathcal{T}_k$, which increases exponentially as time evolves. We will discuss this issue later.

Recall that the state of the POMDP at time step k is $\mathbf{s}_k = (\mathbf{d}_k, \mathbf{v}_k)$. The belief state $\mathbf{b}_k$ is the posterior joint distribution of $\mathbf{d}_k$ and $\mathbf{v}_k$ given H_k, or equivalently, the functions $P_{\mathbf{d}_k|H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$. Therefore, updating the belief state $\mathbf{b}_k$ is equivalent to updating these functions. Moreover, given $\mathbf{y}_k = \mathbf{y}$ and $\mathbf{u}_k = \mathbf{u}$, the functions $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ and $P_{\mathbf{d}_k|H_k}$ can be computed from $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ using Bayes' rule and the law of total probability in the following way:

$$f_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\mathbf{v}|\mathbf{d}, H_k) = \frac{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} g(\mathbf{v}, \mathbf{d}, \boldsymbol{\tau}) P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}, \quad (3)$$

and,

$$P_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k) = \frac{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}{\sum_{\mathbf{e}\in\Omega_s}\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{e}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{e}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}, \quad (4)$$

where $g(\mathbf{v}, \mathbf{d}, \boldsymbol{\tau}) = f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d}, \boldsymbol{\tau}(2), H_k)$. Note that the value $\boldsymbol{\tau}(3)$ is the prior weight value of the starting node $\boldsymbol{\tau}(2)$ for the link represented by the tuple $\boldsymbol{\tau}$. This means that to update the belief state $\mathbf{b}_k$ at time step k, it is sufficient to update functions $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ for all the possible links that the state transition law allows at each time step.

In this paper, we assume a dynamic linear model and a initial Gaussian distribution for $\mathbf{v}_0|\mathbf{d}_0$ given each $\mathbf{d}_0 \in \Omega_s$. Each distribution can be characterized by a density function $f_{\mathbf{v}_0|\mathbf{d}_0}$ with the mean vector $\boldsymbol{\mu}_{\mathbf{d}_0}$ and the covariance matrix $\mathbf{C}_{\mathbf{d}_0}$. Consequently, the density function $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$, $k \geq 1$, will also be Gaussian, and we use $\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ for the mean vector and $\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ for the covariance matrix.

Given the action $\mathbf{u}_k = \mathbf{u}$, the measurements $\mathbf{y}_k = \mathbf{y}$, and the history H_{k-1}, the functions $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ can be computed using the following two steps:

1. For a given any link, represented by a tuple $\boldsymbol{\tau}$ in $\mathcal{T}_k$, the following system of linear equations describes the dynamics of the moving targets along that link:

$$\begin{cases} \mathbf{v}_k = \mathbf{v}_{k-1}, \\ \mathbf{y}_k = \mathbf{A}_{\boldsymbol{\tau}(1)}\mathbf{v}_k + \mathbf{w}_k. \end{cases}$$

In this system, $\mathbf{v}_k \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}, \mathbf{C}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)})$. The vector $\mathbf{v}_{k-1}$ also has a Gaussian distribution, which is represented by the density function $\boldsymbol{\tau}(4)$ in the tuple $\boldsymbol{\tau}$. Therefore, the values $\boldsymbol{\mu}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}$ and $\mathbf{C}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}$ for the given support pair $\mathbf{d}_k = \boldsymbol{\tau}(1)$ and $\mathbf{d}_{k-1} = \boldsymbol{\tau}(2)$ can be computed by using a Kalman filter. Note that for each possible tuple $\boldsymbol{\tau} \in \mathcal{T}_k$, we have to use a separate Kalman filter to update the corresponding prior distribution function $\boldsymbol{\tau}(4)$ of the related link. In the next section, we will discuss the computational issues that usually arise. Once this update is completed, the value of the function $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\boldsymbol{\tau}(1), \boldsymbol{\tau}(2), H_k)$ for any $\mathbf{v} \in \mathbb{R}^s$ can then be computed.

2. For a given tuple $\boldsymbol{\tau} \in \mathcal{T}_k$, the posterior weight value is computed in the following way:

$$P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2), H_k) = \frac{f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\tau}, \boldsymbol{\tau}) f_2(\boldsymbol{\tau}, \boldsymbol{\tau})}{\sum_{\boldsymbol{\nu}\in\mathcal{T}_k} f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\tau}) f_2(\boldsymbol{\nu}, \boldsymbol{\tau})}, \tag{5}$$

where functions $f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\tau})$ and $f_2(\boldsymbol{\nu}, \boldsymbol{\tau})$ are defined as:

$$f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\tau}) = f_{\mathbf{y}_k|\mathbf{d}_k,\mathbf{d}_{k-1},\mathbf{u}_k,H_{k-1}}(\mathbf{y}|\boldsymbol{\nu}(1), \boldsymbol{\tau}(2), \mathbf{u}, H_{k-1}),$$

and,

$$f_2(\boldsymbol{\nu}, \boldsymbol{\tau}) = P_{\mathbf{d}_k|\mathbf{d}_{k-1}}(\boldsymbol{\nu}(1)|\boldsymbol{\tau}(2))\boldsymbol{\tau}(3) = p_{\boldsymbol{\tau}(2)\boldsymbol{\nu}(1)}\boldsymbol{\tau}(3),$$

respectively. Note that the value $p_{\boldsymbol{\tau}(2)\boldsymbol{\nu}(1)}$, that can be computed from the state transition law, is equal to zero for the non-existing links from $\boldsymbol{\tau}(2)$ to $\boldsymbol{\nu}(1)$. Moreover, for the function f_1, it can be easily shown that

$$\mathbf{y}_k|\mathbf{d}_k, \mathbf{d}_{k-1}, \mathbf{u}_k, H_{k-1} \sim \mathcal{N}(\mathbf{A}_{\mathbf{d}_k}\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}, \mathbf{A}_{\mathbf{d}_k}\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}\mathbf{A}'_{\mathbf{d}_k} + \sigma_w^2\mathbf{I}_{l_k}),$$

where $\mathbf{A}'_{\mathbf{d}_k}$ is the transpose of the matrix $\mathbf{A}_{\mathbf{d}_k}$. Note that the values $\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ and $\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ have already been computed in the previous step.

As mentioned before, after going through the above two steps for all the tuples in $\mathcal{T}_k$ and also computing the functions $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ and $P_{\mathbf{d}_k|H_k}$ given any $\mathbf{d}_k \in \Omega_s$ using (3) and (4), the update of the belief state $\mathbf{b}_k$ is completed.

B. COMPUTATIONAL ISSUES IN THE BELIEF STATE UPDATE

The belief state update procedure described in the previous section shows the computations required once measurements are received at each time step. To highlight the computational issues, consider the extreme case where the transition law $P_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ allows the targets to move freely to any position. Also, assume a uniform initial support density function $P_{\mathbf{d}_0}$ over Ω_s. Knowing that $|\Omega_s| = N^s$, then for each support candidate $\mathbf{d}_0$ in Ω_s, there would be N^s possibilities for the support $\mathbf{d}_1$. Therefore, N^{2s} Kalman filters must be employed to update the belief state at time $k = 1$. Extending this line of thought, at any time k, $N^{(k+1)s}$ Kalman filters are needed which is an exponential increase in computations over time.

Of course in reality, the assumed transition law $P_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ does not provide such freedom for the moving target(s). Although this reduces the number of possible transitions, the exponential increase in computation remains. This is seen in the following example. Again assume, at time $k = 1$, a uniform prior distribution $P_{\mathbf{d}_0}$ over Ω_s. However, now assume the transition law restricts the number of possible transitions for each support candidate to $q \leq N^s$. Following the same argument as above, there are now $N^s q$ links involved in the belief state update at time $k = 1$ and for time $k = 2$, $N^s q^2$ links are present. Similarly, $N^s q^k$ links (equivalently Kalman filters) are required for the belief state update at any time step k.

Obviously, updating the belief state according to the above procedure requires a vast amount of time and also computational power. To deal with the computational volume we introduce two techniques explained below. This allows us to examine the proposed solution methods to Problems 1 and 2 and avoid some of the related computational load.

As we have shown, propagating the entire distribution quickly becomes unmanageable due to its exponential growth-rate in terms of the number of possible links. This issue is also a major problem in the context of data association in the multi-target tracking problem. In this problem, the main question asked is: At each time step, which target does of the collected measurements at that time step represent? If all the observations-to-target associations were considered valid, a similar expansion of the distribution representing the current target states would occur.

We can consider the procedure in Section II-A as one of associating the measurements $\mathbf{y}_k$ to the *possible* support-value pairs, which are characterized by their corresponding belief state components $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$. The similarity to the data association problem motivated the adaptation of available solutions in the target tracking context. We have implemented two separate methods based on popular data association algorithms to combat the expansion of our belief state distributions, namely: multi-hypothesis tracking (MHT) (see [41]) and joint probabilistic data association (JPDA) (see [39] and [40]).

Algorithm 1: The essential question to ask is how to choose or approximate the proper Gaussian distributions and weights to represent different possible support candidates along different possible routes. There are several approaches to accomplish this, e.g., MMSE, MLE, etc., and in the target tracking literature, there are also numerous implementations applying these approximations. We model our first method after MHT.

The basis of MHT lies in multi-hypothesis testing wherein a choice must be made between several concurrent hypotheses based on a sufficient statistic. Given a set of J hypotheses $\mathcal{H} = \{h_i\}_{i=1}^J$ and the measurements $\mathbf{y}_k = \mathbf{y}$, the simplest

version of multi-hypothesis testing chooses one hypothesis h^* based on finding an optimal solution for the following optimization problem:

$$h^* = \arg \max_{1 \le i \le J} \{\mathcal{L}(h_i|\mathbf{y})\, P(h_i)\},$$

where $\mathcal{L}(h_i|\mathbf{y})$ and $P(h_i)$ denote the likelihood given the measurement $\mathbf{y}$ and the prior probability of the hypothesis h_i, respectively. The function $\mathcal{L}(h_i|\mathbf{y})$ is equivalent to $f_{\mathbf{y}_k|h}(\mathbf{y}|h_i)$, the conditional probability density function of the measurement vector $\mathbf{y}_k$ conditioned on the hypothesis h. Therefore, the above optimization problem will be equal to

$$h^* = \arg \max_{1 \le i \le J} \left\{f_{\mathbf{y}_k,h}(\mathbf{y}, h_i)\right\}.$$

In our setting, similar to the target tracking setting, more measurements accrue as time progresses, providing more information about the underlying state. Obviously, more information can help increase the accuracy of the association process. Even in dynamic systems, observations of future states provide valuable information about the past system states. This is the basic motivation behind the MHT method.

Consider a time-varying signal estimation problem with a finite horizon of length κ. At each time step, a measurement is acquired through the observation law. Then, the most accurate selection of a hypothesis is made by collecting all of the measurements and maintaining each possible association over the entire horizon. This results in a hypothesis tree, κ levels deep. Each level consists of nodes representing the possible support-value pairs (supports may be repeated) and the corresponding weights $P_{\mathbf{d}_k|H_k}$ for each time step k. At the final time step $k = \kappa$, applying a multi-hypothesis test to the set of leaf nodes selects the leaf terminating the most probable route in the tree (most probable with respect the posterior distribution). This selection not only yields a current estimate of the support and values that belong to the leaf at the end of the route, but this route, itself, describes the most probable sequence of support-value pairs. Note, in our setting, the value component $\mathbf{v}_k$ of the state has a static transition law, and thus the final estimate of $\mathbf{v}_\kappa$ would be the more accurate that the estimates of $\mathbf{v}_k, k < \kappa$ because they would not have benefited from the later observations.

Let $\boldsymbol{\rho}_k^{(i)}$ be an ordered k-tuple representing the ith possible route at time k. The first component $\boldsymbol{\rho}_k^{(i)}(1)$ of this tuple is the current node of this route and the last component $\boldsymbol{\rho}_k^{(i)}(k)$ is the initial node of this route. For example, for the selected route in Fig. 1, which we refer to as the ith route, we have the following tuples for time steps $k = 1, 2$, and $k = 3$, respectively: $\boldsymbol{\rho}_1^{(i)} = \mathbf{e}_3$, $\boldsymbol{\rho}_2^{(i)} = (\mathbf{e}_2, \mathbf{e}_3)$, and $\boldsymbol{\rho}_3^{(i)} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$. Note that at time step k, the total number of possible routes is equal to $|\mathcal{T}_k|$. Let $h_i^{(k)}$ be the hypothesis that at time step k, the ith route is the true route. If this hypothesis is true, then this means that $\mathbf{d}_k = \boldsymbol{\rho}_k^{(i)}(1), \mathbf{d}_{k-1} = \boldsymbol{\rho}_k^{(i)}(2), \ldots, \mathbf{d}_1 = \boldsymbol{\rho}_k^{(i)}(k)$. Additionally, let $h^{(k)}$ be the random variable representing the true hypothesis; $h^{(k)}$ takes values on set of all hypotheses $h_i^{(k)}$, $i = 1, \ldots, |\mathcal{T}_k|$, at time step k.

Determining the terminal node of the probable route is done by ranking the possible routes with their *track scores*. Let $\boldsymbol{o}_k$ be a tuple defined as $\boldsymbol{o}_k = (\mathbf{y}_1, \ldots, \mathbf{y}_k)$. Then, given the tuple $\boldsymbol{o}_\kappa = \boldsymbol{o}$, where $\boldsymbol{o} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(\kappa)})$, and the history $H_\kappa = \{\mathbf{u}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{u}^{(\kappa)}, \mathbf{y}^{(\kappa)}\}$ at time step κ, the track score $S_{\boldsymbol{\rho}_\kappa^{(i)}}$ of the route i, represented by the tuple $\boldsymbol{\rho}_\kappa^{(i)}$, is defined as

$$S_{\boldsymbol{\rho}_\kappa^{(i)}} = \mathcal{L}(h_i^{(\kappa)}|\boldsymbol{o})P(h_i^{(\kappa)}) = f_{\boldsymbol{o}_\kappa,h^{(\kappa)}}(\boldsymbol{o}, h_i^{(\kappa)}),$$

where $f_{\boldsymbol{o}_\kappa,h^{(\kappa)}}$ is the joint probability density function of $\boldsymbol{o}_\kappa$ and $h^{(\kappa)}$ at time step κ. By expanding the function $f_{\boldsymbol{o}_\kappa,h^{(\kappa)}}$, the value of $S_{\boldsymbol{\rho}_\kappa^{(i)}}$ is equal to

$$\begin{aligned} &f_{\mathbf{y}_\kappa|\mathbf{d}_\kappa,\mathbf{d}_{\kappa-1},\mathbf{u}_\kappa,H_{\kappa-1}}(\boldsymbol{o}(\kappa)|\boldsymbol{\rho}_\kappa^{(i)}(1), \boldsymbol{\rho}_\kappa^{(i)}(2), \mathbf{u}^{(\kappa)}, H_{\kappa-1}) \\ &\quad \times p_{\boldsymbol{\rho}_\kappa^{(i)}(2)\boldsymbol{\rho}_\kappa^{(i)}(1)} P_{\mathbf{d}_{\kappa-1}|\mathbf{d}_{\kappa-2},H_{\kappa-1}}(\boldsymbol{\rho}_\kappa^{(i)}(2)|\boldsymbol{\rho}_\kappa^{(i)}(3), H_{\kappa-1}). \end{aligned}$$

Note that all of the components in this expression can be computed using the equations presented in Section II-A. Looking at (5), one can conclude that $S_{\boldsymbol{\rho}_\kappa^{(i)}}$ is proportional to $S^{\text{eq.}}_{\boldsymbol{\rho}_\kappa^{(i)}}$ where

$$S^{\text{eq.}}_{\boldsymbol{\rho}_\kappa^{(i)}} = \prod_{k=1}^{\kappa} t(\boldsymbol{o}, \boldsymbol{\rho}_k^{(i)}, H_k, k),$$

and the function t is defined as

$$t(\boldsymbol{o}, \boldsymbol{\rho}_k^{(i)}, H_k, k) = \begin{cases} f_{\mathbf{y}_k|\mathbf{d}_k,\mathbf{d}_{k-1},\mathbf{u}_k,H_{k-1}}(\boldsymbol{o}(k)|\boldsymbol{\rho}_k^{(i)}(1), \boldsymbol{\rho}_k^{(i)}(2), \mathbf{u}^{(k)}, H_{k-1}) \\ \qquad \times p_{\boldsymbol{\rho}_k^{(i)}(2)\boldsymbol{\rho}_k^{(i)}(1)}, & k \ge 2, \\ P_{\mathbf{d}_1|H_1}(\boldsymbol{\rho}_1^{(i)}(1)|H_1), & k = 1. \end{cases}$$

Therefore, we can use $S^{\text{eq.}}_{\boldsymbol{\rho}_\kappa^{(i)}}$ equivalently to find most probable route at time step κ. Practical concerns such as numerical round-off errors motivate a conversion of $S^{\text{eq.}}_{\boldsymbol{\rho}_\kappa^{(i)}}$ to a logarithmic representation, $S^{\log}_{\boldsymbol{\rho}_\kappa^{(i)}}$, in the following way

$$S^{\log}_{\boldsymbol{\rho}_\kappa^{(i)}} = \sum_{k=1}^{\kappa} \log\left(t(\boldsymbol{o}, \boldsymbol{\rho}_k^{(i)}, H_k, k)\right).$$

This alternative representation also provides a simple recursive calculation,

$$S^{\log}_{\boldsymbol{\rho}_k^{(i)}} = S^{\log}_{\boldsymbol{\rho}_{k-1}^{(i)}} + \log\left(t(\boldsymbol{o}, \boldsymbol{\rho}_k^{(i)}, H_k, k)\right), \qquad k = 2, \ldots, \kappa.$$

As mentioned earlier, when no approximation techniques are used, the number of possible routes increases exponentially as time evolves. To deal with this problem, most MHT implementations use a sliding window to designate when the decision making process begins. The sliding window heuristic limits the size of the hypothesis tree that must be maintained. Consider a sliding window of size w. After the first w time steps, a hypothesis test is performed, selecting the current leaf with the maximum track score. The route terminating with the selected leaf is then traced back w levels in the tree. The node at the base of this branch and all of its descendants are retained, while all other nodes and branches from time-step $(k - w)$ onward are discarded. At every time step $k \ge w$, a similar pruning of the tree is performed. The larger the sliding

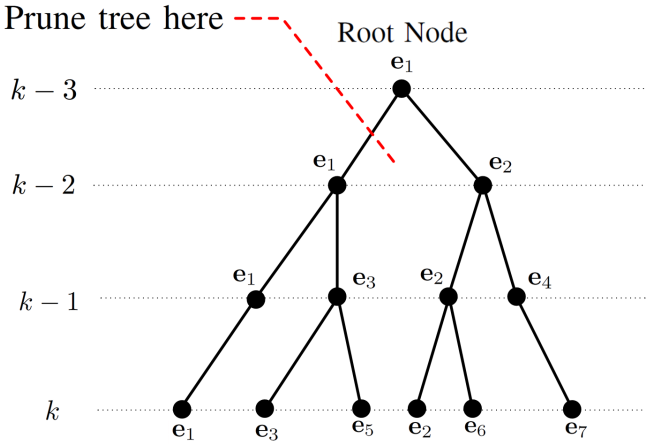


FIGURE 2. This illustration (adapted from [41]) represents a possible hypothesis tree for a time-varying signal scenario. Identically labeled nodes indicate that the support has remained the same over the transition. A sliding window of size $w = 2$ is depicted and the pruning action has indicated that the maximum track score belongs to the route represented by the tuple $(\mathbf{e}_6, \mathbf{e}_2, \mathbf{e}_2, \mathbf{e}_1)$. Thus, the pruning is done as shown. This has reduced the computation for time step $k + 1$ by 50%.

window the better the chances of a correct association due to the increased information accumulated. Fig. 2 illustrates this process for an arbitrary time step.

The sliding window size must be balanced with the computational time and memory constraints. The maximum reduction in computation from our MHT based algorithm would be a sliding window of size one. Thus, a hypothesis test would be performed at every time step. One data-association technique that uses a sliding window of size one is JPDA (see [39] and [40]). This is the basis for Algorithm 2, described next.

Algorithm 2: Consider the set $\mathcal{T}_k$ defined in Section II-A. Looking at the tuples in this set, each of which is associated with a possible link, we can find multiple tuples that all share a similar ending node. For example, in Fig. 1, there are 4 instances for node $\mathbf{e}_1$ at time step $k = 3$. To cope with the computational volume we combine the posterior weights and distributions attached to different instances of an ending node. More specifically, at each time step k, instead of keeping all instances of a support candidate $\mathbf{d} \in \Omega_s$, we represent the candidate with only one *approximate* posterior weight value $\hat{P}_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k)$ and one *approximate* posterior distribution function $\hat{f}_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\cdot|\mathbf{d}, H_k)$. This will force each set $\mathcal{T}_k$ for $k = 1, 2, \ldots$, to have a fixed cardinality equal to the cardinality of the set Ω_s, i.e., $|\mathcal{T}_k| = N^s$.

To pursue this, consider (3). Define $\alpha_{\boldsymbol{\tau},\mathbf{d},H_k}$ to be

$$\alpha_{\boldsymbol{\tau},\mathbf{d},H_k} = \frac{P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}.$$

Then, equation (3) simplifies to

$$f_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\mathbf{v}|\mathbf{d}, H_k) = \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d}, \boldsymbol{\tau}(2), H_k).$$

This is the representation of a Gaussian mixture with weight values $\alpha_{\boldsymbol{\tau},\mathbf{d},H_k}$ and Gaussian components $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$. Given $\mathbf{d}_k = \mathbf{d}$ at time step k, the mean $\boldsymbol{\mu}_{\mathbf{d}}$ of this mixture will be equal to

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{d}} &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \int_{\mathbf{v}} \mathbf{v} f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d}, \boldsymbol{\tau}(2), H_k) d\mathbf{v} \\ &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \boldsymbol{\mu}_{\mathbf{d}|\boldsymbol{\tau}(2)}.\end{aligned}$$

Also, the covariance matrix $\mathbf{C}_{\mathbf{d}}$ of this mixture can be computed in the following way:

$$\begin{aligned}\mathbf{C}_{\mathbf{d}} &= \mathbf{E}\left[(\mathbf{v}_k|\mathbf{d}, H_k)(\mathbf{v}_k|\mathbf{d}, H_k)'\right] - \boldsymbol{\mu}_{\mathbf{d}}\boldsymbol{\mu}_{\mathbf{d}}' \\ &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \\ &\quad \times \int_{\mathbf{v}} \mathbf{v}\mathbf{v}' f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d}, \boldsymbol{\tau}(2), H_k) d\mathbf{v} - \boldsymbol{\mu}_{\mathbf{d}}\boldsymbol{\mu}_{\mathbf{d}}' \\ &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \left(\mathbf{C}_{\mathbf{d}|\boldsymbol{\tau}(2)} + \boldsymbol{\mu}_{\mathbf{d}|\boldsymbol{\tau}(2)}\boldsymbol{\mu}_{\mathbf{d}|\boldsymbol{\tau}(2)}'\right) - \boldsymbol{\mu}_{\mathbf{d}}\boldsymbol{\mu}_{\mathbf{d}}'.\end{aligned}$$

Now, we approximate the Gaussian mixture density function with a regular Gaussian density function $\hat{f}_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\cdot|\mathbf{d}, H_k)$ that has the mean $\boldsymbol{\mu}_{\mathbf{d}}$ and the covariance matrix $\mathbf{C}_{\mathbf{d}}$. We apply this approximation to all the support candidates in Ω_s.

To further decrease the computation volume, we first compute $P_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k)$ for all possible support values $\mathbf{d} \in \Omega_s$ using (4). We then prune this probability function by keeping a certain number h of the weights with the highest probability values in $P_{\mathbf{d}_k|H_k}$ and forcing the rest to be zero. Adjusting the nonzero values to sum to one will result in having an approximate probability function $\hat{P}_{\mathbf{d}_k|H_k}$ to use as the prior at time step $k + 1$.

This means that at each time step, we will only have h initial nodes instead of N^s nodes. Therefore, at time step k, instead of using $N^s q^k$ Kalman filters for the scenario explained above, only hq Kalman filters are required to update the belief state. This will decrease the amount of computation significantly.

As mentioned earlier, the ideas used in Algorithm 2 are motivated by a target tracking data association heuristic algorithm known as JPDA. We refer the interested reader to [39] and [40] to learn more about this heuristic.

III. ROLLOUT

In both problems introduced in Section I, we want to make optimal decisions at each time step (based on appropriate criteria) by either selecting the measurement matrix $\mathbf{A}_k$ or choosing the number of measurements l_k, given the history H_k. In Section II, we formulated these problems as POMDPs. In this section, we describe our solution approach based on an approximation method called *rollout*. Our description relies heavily on well established ideas and terminology from POMDP theory, which are readily available in [30] and [31].

In principle, the solution to a POMDP problem is, at each time k, a mapping that takes the history H_k and gives an

optimal action from $\mathcal{A}$. However, POMDP theory establishes that is suffices to find an optimal mapping $\pi_k^* : \mathcal{B} \rightarrow \mathcal{A}$ for $k = 1, \ldots, m$, where $\mathcal{B}$ is the set of distributions over the state space $\mathcal{S}$, and $\mathcal{A}$ is the actions space. If all actions are chosen using these optimal mappings, then at time $k = m$, the predefined objective function is optimized and the resulting *optimal policy* $\pi^* = \{\pi_1^*, \ldots, \pi_m^*\}$ has been generated.

We refer to the objective function as the *expected cumulative cost*. Given a policy $\pi = \{\pi_1, \ldots, \pi_m\}$ the expected cumulative cost is defined as

$$V_m^{\pi}(\mathbf{b}_1) = \mathbf{E}\left[\sum_{k=1}^{m} C_k(\mathbf{b}_k, \pi_k(\mathbf{b}_k)) \middle| \mathbf{b}_1\right]. \tag{6}$$

Subsequently, the optimal objective function value $V_m^{\pi^*}$ is achieved when $\pi = \pi^*$ in (6).

By applying *Bellman's principle* to (6), the optimal objective function $V_m^{\pi^*}$ and the optimal policy π^* can be characterized by the following two equations:

$$V_m^{\pi^*}(\mathbf{b}_1) = \min_{\mathbf{u}}(C_1(\mathbf{b}_1, \mathbf{u}) + \mathbf{E}[V_{m-1}^{\pi^*}(\mathbf{b}_2)|\mathbf{b}_1, \mathbf{u}]), \tag{7}$$

and

$$\pi_1^*(\mathbf{b}_1) = \arg\min_{\mathbf{u}}(C_1(\mathbf{b}_1, \mathbf{u}) + \mathbf{E}[V_{m-1}^{\pi^*}(\mathbf{b}_2)|\mathbf{b}_1, \mathbf{u}]). \tag{8}$$

Let

$$Q_{m-k}(\mathbf{b}_k, \mathbf{u}) = C_k(\mathbf{b}_k, \mathbf{u}) + \mathbf{E}[V_{m-k}^{\pi^*}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}] \tag{9}$$

be the *Q-value* of taking action $\mathbf{u}$ at belief state $\mathbf{b}_k$. Combining (8) and (9), we can now view the optimal action at time k as

$$\pi_k^*(\mathbf{b}_k) = \arg\min_{\mathbf{u}} Q_{m-k}(\mathbf{b}_k, \mathbf{u}).$$

In other words, the optimal action can be found at any time step k by identifying the action with the minimum Q-value at belief state $\mathbf{b}_k$.

The two summands in (9), $C_k(\mathbf{b}_k, \mathbf{u})$ and $\mathbf{E}[V_{m-k}^{\pi^*}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}]$, are the immediate cost incurred (by $\mathbf{u}$) and the *expected cost-to-go* (ECTG) at the belief state $\mathbf{b}_k$, respectively. This breakdown inspires a natural separation in action selection approaches, namely *myopic* and *non-myopic*. Myopic action selection only considers the immediate impact (cost) of the action $\mathbf{u}$ at belief state $\mathbf{b}_k$. These approximations yield a *myopic* policy $\hat{\pi}$ which selects actions by

$$\hat{\pi}_k(\mathbf{b}_k) = \arg\min_{\mathbf{u}} C_k(\mathbf{b}_k, \mathbf{u}).$$

In relation to the Q-values, the myopic policy can be viewed as replacing the ECTG with a fixed constant, which can be assumed to be zero without loss of generality. In contrast, non-myopic methods attempt to approximate the ECTG term of the Q-value. If the estimation of the ECTG is suitable, it will evince the impact of the current action on the future costs to be incurred. There are several approaches for Q-value approximation [43]. We choose to implement the *rollout* method.

Rollout replaces the optimal policy used in the ECTG with a so called *base policy* π^{base}, creating the following Q-value approximation:

$$\tilde{Q}_{m-k}(\mathbf{b}_k, \mathbf{u}) = C_k(\mathbf{b}_k, \mathbf{u}) + \mathbf{E}[V_{m-k}^{\pi^{\text{base}}}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}].$$

This approximation differentiates the effects of the actions not only at the current time step but over the future horizon, providing *multi-step lookahead* as opposed to the *1-step lookahead* of the myopic policy. It should also be noted that the simulation results presented in the next section produced by rollout exploit *receding horizon control*, replacing the remaining horizon $m - k$ with a fixed value ϑ. This "artificial" horizon is regarded as the *rollout horizon*; rollout defines a ϑ-lookahead policy.

The technical details justifying both receding horizon control and rollout can be found in [43], but we provide the reader with a meaningful result which motivates the use of rollout. By ranking actions over time with their approximate Q-values, rollout produces a policy $\tilde{\pi} = \{\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_m\}$, where

$$\tilde{\pi}_k(\mathbf{b}_k) = \arg\min_{\mathbf{u}} \tilde{Q}_{\vartheta}(\mathbf{b}_k, \mathbf{u}).$$

The policy $\tilde{\pi}$ is guaranteed to perform at least as well as the base policy π^{base} with respect to the objective function. In fact, it is common for rollout to out-perform its base policy. This is related to the fundamental relation of rollout to policy improvement (see [46]).

The previous two approximations methods, rollout and receding horizon control, are used throughout the simulations in Section IV. However, in Section IV-D we apply an additional approximation when Algorithm 1 is employed for data association in the multiple target scenarios. This approximation further limits the amount of computation needed for the Q-value estimation, resembling *completely observable rollout* from [43]. Like completely observable rollout, we endeavour to reduce the computations in the simulations of the future via the reduction of the (simulated) belief state representations. Completely observable rollout accomplishes this by substituting a Dirac delta distribution for the belief states in the rollout approximation of the Q-value and thus applies a base policy which maps the underlying states to actions. This is an "extreme" approximation of the belief state, but it fairs well in certain situations where such base policies are readily available and easily computed. Our approximation is less drastic than that of completely observable rollout, but it still takes advantage of the knowledge of the underlying system state during the Monte Carlo simulations used for the Q-value approximations.

Recall Algorithm 1 and its hypothesis tree from Section II-B. Although expansion of the hypothesis tree is constrained by the choice of the sliding window size, a large portion of computation is still devoted to the prediction step. During the prediction step the tree must be extended to the next level of nodes representing all possible supports at the pending time step. Furthermore, after this expansion, a Kalman filter update computation is required for each new

node in the tree. Upon the culmination of these calculations, the tree is immediately pruned. The pruning process results in a substantial portion of the new nodes being discarded and thus the associated calculations are inconsequential over the remaining time horizon of the scenario. The realization of the limited amount of time, from prediction to pruning, over which these computations are viable motivates our approximation of the belief state for the rollout simulations.

Our belief state reduction, inside rollout, eliminates the computations incurred by this cycle of prediction and pruning. This is accomplished by extending only the tree with nodes representing the best case, where the selected leaf is no longer necessarily the most probable leaf node, but actually represents the ''true'' support-value pair descending from the ''true'' route of the sample signal. This is a viable option within rollout since we can access these details for each sample signal in the Monte Carlo simulation. Given this knowledge and a window size of w, at each time step in the ϑ-step rollout horizon we identify the node representing the true route in the tree and trace the route back $w-1$ time steps yielding the routes *initial node*. Then only the $(w-1)$-generation leaves of the initial node are used in the prediction step, resulting in fewer new leaves and fewer Kalman filter updates. In fact, the new leaves are equivalently those which would have remained after the pruning step if the most probable leaf-node, i.e., the one selected by Algorithm 1, was the true representation of the sample signal. Therefore, we also eliminate the pruning step resulting in a further reduction of computation load.

This extended discussion on Q-value approximation shows that there are obvious trade-offs between myopic and non-myopic policies. The clear benefit of myopic policies is the relatively simple computations of $C_k(\mathbf{b}_k, \mathbf{u})$ required for their *greedy* action choices, based solely on the immediate cost of an action at the current belief state. Often, this alone motivates the implementation of the myopic policy. Conversely, non-myopic approaches must contend with the computational complexity originating from the inclusion of the ECTG term. In particular, rollout must compute the expected cumulative cost of the chosen base policy over $\vartheta - 1$ time steps, and this computation is typically done using Monte Carlo sampling. However, given a scenario where the current action choice substantially impacts the future outlook, rollout provides a distinct advantage due to its lookahead property. The results in the following section empirically compare and contrast the greedy-type policies with rollout.

IV. SIMULATION RESULTS

In this section, we present numerical examples for our solutions to the problems introduced in Section I. To better demonstrate the value of adaptivity and in particular multi-step lookahead policies, we consider several scenarios. The first simulation compares non-adaptive methods to adaptive methods, while the remaining simulations compare several adaptive techniques. Since each simulation contains a significant amount of details about its scenario, we briefly explain the goals and the results of running each simulation.

Our first simulation, Simulation 1, considers a very simple case of a steady state 1-sparse signal with a highly informative prior distribution on the location of the nonzero entry. We find solutions for Problem 1 and evaluate the performance of adaptive methods compared to non-adaptive methods. The results of this simulation indicate that even in such a simple scenario, the proposed adaptive methods perform better than the non-adaptive methods. However, multi-step lookahead does not provide a significant advantage over the 1-step lookahead because of the simplicity of the scenario.

In Simulation 2, an instance of Problem 1, there is a single moving target, i.e., a dynamic 1-sparse signal, that changes its location in an environment with occlusions. When the target occupies a position with an occlusion, the measurements are just noise samples. We assume a uniform prior distribution on the supports, which is less informative than the prior in Simulation 1. These results again show little-to-no difference between the performance of 1-step and multi-step lookahead policies. Even with the augmentation of the problem with occlusions and dynamics, multi-step lookahead does not amass extra *useful* information compared to the 1-step lookahead solution. This relates to the POMDP cost and action space definitions. The simulations based on Problem 2 explicate this further.

In Simulation 3, we use the same scenario as Simulation 2 but we consider solving Problem 2 instead, using a different POMDP action space and cost. Based on this cost, each method must decide between using more measurements or a reduction in support identification accuracy at each time step. The results of this simulation shows a larger performance advantage for multi-step lookahead compared to 1-step lookahead.

Simulation 4 considers Problem 2 with two moving targets in a space with occlusions and assumes semi-informative prior distribution on the supports. Algorithms 1 and 2 from Section II-B are applied to maintain reduced computations for data association. Again, given the performance criterion and the action space of the problem, the multi-step lookahead outperforms the 1-step lookahead.

In Simulations 1 and 2, we use two libraries of waveforms to build the measurement matrices. The waveforms in each library build a *Grassmannian line packing* (see [47] and [48]) in the space $\mathbb{R}^N$. These libraries share a common property which inspired their use in our simulations; For the given number of waveforms in each respective library, every pair of waveforms in that library achieves a maximum angular distance from one another. This means that the waveforms in each library ''slice'' the sparse signal space $\mathbb{R}^N$ as equally as possible. Therefore, by using a fraction of these waveforms as rows of our measurement matrices, we are searching for the sparse signal in a subspace of $\mathbb{R}^N$ in a semi-uniform manner.

In Simulations 3 and 4, we use a method that was originally introduced in [18] to construct the measurement matrix once the number of measurements is chosen at a particular time step. In this method, the function $P_{\mathbf{d}_k|H_k}$ at time step k is used

to determine the matrix entries for that time step. This method has also motivated us to implement a similar adaptive scheme in Simulation 2, referred to as the "Bhattacharyya Distance" policy, in which once we build the measurement matrix library from the Grassmannian line packing, we compute a "characterizing" function that specifies how the "energy" of the matrix is distributed along its columns. We then choose the matrix whose energy function has the minimum Bhattacharyya distance to $P_{\mathbf{d}_k|H_k}$ at time step k as the measurement matrix.

A. SIMULATION 1: 1-SPARSE STATIC SIGNAL (PROBLEM 1)

In this simulation, we consider a simple static scenario. We assume that the signal $\mathbf{x}_k$ is 1-sparse in $\mathbb{R}^{75}$ and that its nonzero entry stays in the same location at all times. We further assume a specific prior distribution on the support of $\mathbf{x}_k$. The probability mass function (pmf) $P_{\mathbf{d}_0}$ corresponding to this prior is shown in Fig. 3. This prior indicates the nonzero entry of the signal is located somewhere in the first 16 indices of the signal. This simulation utilizes 100 signal samples and we rerun the experiment 50 times for each sample. Each signal sample is created by drawing a sample from the prior pmf $P_{\mathbf{d}_0}$ for the locations of the target and then drawing samples from a $\mathcal{N}(0, \sigma^2)$ distribution for the amplitudes.

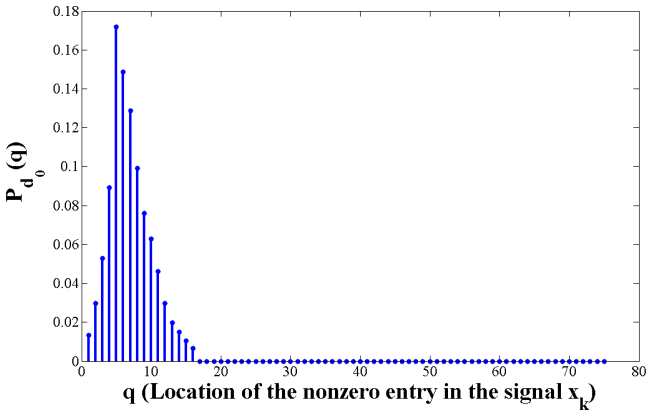


FIGURE 3. Prior structure used for $P_{\mathbf{d}_0}$ in Simulation 1.

We consider an instance of Problem 1 in this simulation where the number l_k of measurements per step is set to one, and the total time steps (total number of measurements) is $m = 8$. We compare five different (three non-adaptive and two adaptive) methods:

1. Random: In this method, $\mathbf{A}_k$ is a matrix where all its $(l_k \times N)$ entries are i.i.d. samples from a Gaussian distribution $\mathcal{N}(0, 1/N)$. We also normalize the rows of each matrix $\mathbf{A}_k$.

2. Limited Random: Knowing the prior $P_{\mathbf{d}_0}$, when assembling rows of the measurement matrix $\mathbf{A}_k$, we divide each measurement row $\mathbf{A}_k(i)$ into two parts: the first part is a 16-dimensional vector, and the second part is a 59-dimensional vector. We fill the first 16 entries with i.i.d. random Gaussian samples and normalize the resulting vector to have norm one. We fill the second part with zeros. This policy relies on knowing the support of the prior, in contrast to the first policy, which does not exploit this information. This policy is to explore the benefits of using a predefined library of vectors that are known to be "good," in contrast to random vectors.

3. Random from Library: Similar to Limited Random, each row of the measurement matrix $\mathbf{A}_k$ is broken into two parts, where the second part is a 59-dimensional vector of zeros. However, the first part of the rows are chosen randomly from a static library of 50 measurement vectors that together, constitute a Grassmannian line packing (see [47] and [48]) in $\mathbb{R}^{16}$.

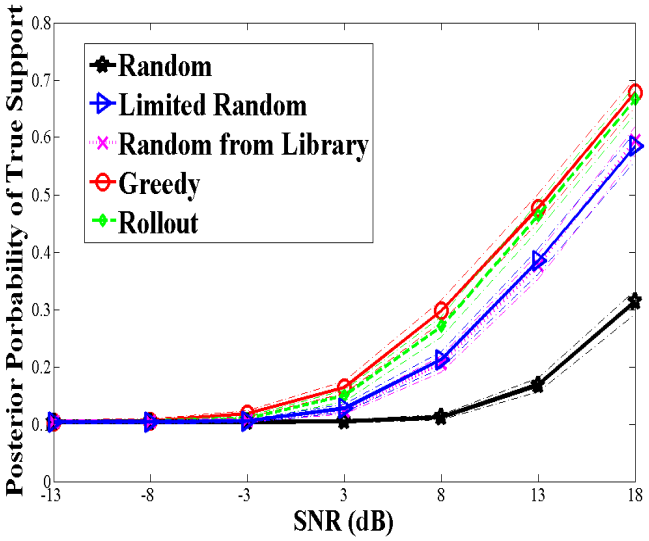
4. Greedy: In this approach, we also break the measurement rows into two parts like methods 2 and 3 and we determine the first 16-dimensional part of the vector. We set the decision-making horizon to 1, i.e., at each time step k, we choose the best vector from the Grassmannian library that minimizes the 1-step ahead belief cost $C_k(\mathbf{b}_k, \mathbf{u}_k)$. Therefore the actions are chosen in a *greedy* manner.

5. Rollout: This policy is based on the rollout method in Section III, in which at each time step and for each action candidate from the Grassmannian library, the Q-value is approximated over a four step rollout horizon. The estimated Q-value, for each candidate action, is the average of 50 (four step) Q-value samples. The action with the minimum Q-value approximation is selected. The base policy for Rollout is the Random from Library method.

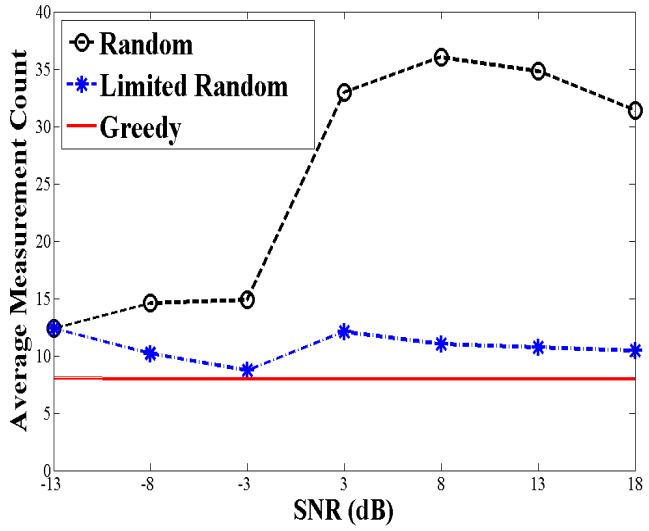
Fig. 4(a) shows the performance of the five methods introduced above. The metric used in this figure for comparing these methods is the posterior probability of the true support after $m = 8$ measurements, i.e., the value of $P_{\mathbf{d}_8|H_8}(\mathbf{d}_T|H_8)$, where $\mathbf{d}_T$ is the true location of the nonzero entry of the signal. We have shown the performance of these methods for different values of signal-to-noise-ratio (SNR), which is defined as SNR $= \sigma^2/\sigma_w^2$. This figure shows that variations of POMDP, i.e., Greedy and Rollout, perform similar to each other in this simple static scenario, but both perform better than the three non-adaptive methods at moderate and high SNR. At low SNR all methods perform similarly.

A second experiment is run to see, on average, how many more measurements Random and Limited Random require in order to reach the performance of Greedy when the same metric, $P_{\mathbf{d}_8|H_8}(\mathbf{d}_T|H_8)$, is used for comparison. Fig. 4(b) shows the results for different values of SNR. The number of measurements for Greedy is set to $m = 8$ for all SNRs. The plots suggest that in this simple static scenario with the particular prior used, simply knowing that the true support lies within the first 16 indices provides significant performance gains over the random scheme. Moreover, adaptation only provides marginal gains over methods that exploit the highly informative prior.

We anticipate similar conclusions in the context of Problem 2. That is, due to the simple and static nature of the problem, multi-step lookahead does not offer much advantage over the single-step lookahead. Also, we anticipate that



(a)



(b)

FIGURE 4. **(a) Performance comparison of all methods in Simulation 1. The dashed lines indicate the 95% confidence intervals. (b) Average number of measurements required for Random and Limited Random to reach the performance of Greedy in Simulation 1.**

adaptivity would again offer minimal gain over the Limited Random method, which exploits the highly informative prior used in this scenario.

B. SIMULATION 2: 1-SPARSE TIME-VARYING SIGNAL (PROBLEM 1 WITH OCCLUSIONS)

We consider a 1-sparse dynamic signal in $\mathbb{R}^{20}$ for this simulation. Differing from Simulation 1, we assume a uniform prior for $P_{\mathbf{d}_0}$ and the presence of occlusions in certain areas. When the signal support corresponds to occluded locations, the received observations are simply noise samples, i.e., $\mathbf{y}_k = \mathbf{w}_k$. The occlusions for Simulation 2 are located at indices 5, 6, 11, and 12. The movement of the target is modeled by the transition law shown in Fig. 5. The initial position of the true target is located at the second index and the target moves one position to the right until reaching index 20. At this point, the target begins shifting in the reverse direction back to position one, where the target will again reverse its direction of movement. The horizon of Simulation 2 is 40 time steps, so the target will reverse direction twice. Furthermore, the target will be occluded at times $k = \{4, 5, 10, 11, 27, 28, 33, 34\}$. Finally, we set SNR $= 14$ (dB) and run the experiment 36 times for both the Greedy and Rollout and 300 times for the both the Random from Library and the Bhattacharyya Distance algorithms that we define below.

Similar to Simulation 1, the scenario of interest is based on Problem 1 where $l_k = 5$. The policies compared in this set of simulations consists of one non-adaptive and three adaptive schemes. The action selection is again among a set of measurement matrices. To build this set, we first import a Grassmannian packing consisting of 16 vectors in $\mathbb{R}^{16}$. The 16 vectors that build this Grassmannian packing are then combined into different sets consisting of five vectors. Note that we are considering every possible combination (without repeats) of 5 vectors among 16 vectors, resulting in 4368 sets of vectors. Using the vectors in each set, we build a (5×16) matrix. These matrices are then augmented by inserting four columns of zeros in the respective occlusion positions. These 4368 matrices now comprise the library of compressive measurement matrices.

Due to this large action space we developed a heuristic to restrict the set of possible actions based on the prior distribution on the support at each time step. The restricted sets of actions are generated in advance by a clustering algorithm know as *K-medoids* (see [49]). To begin the clustering process, we form a power-vector to accompany each matrix in the library. These power-vectors help us "classify" the matrices. The power-vector $\boldsymbol{\phi}_{\mathbf{A}}$ of a matrix $\mathbf{A}$ is a (20×1) vector where its jth component $\boldsymbol{\phi}_{\mathbf{A}}(j)$ is defined as

$$\boldsymbol{\phi}_{\mathbf{A}}(j) = \frac{\sum_{i=1}^{5} |\mathbf{A}(i)(j)|}{\sum_{j=1}^{20} \sum_{i=1}^{5} |\mathbf{A}(i)(j)|},$$

where $\mathbf{A}(i)(j)$ is the entry of the matrix $\mathbf{A}$ located at its ith row and its jth column. These power-vectors portray how the power of the matrix is "distributed" over its different columns and resemble probability mass functions. The matrix library is then sorted into 400 clusters by the K-medoids algorithm, using the Bhattacharyya distance to classify the measurement matrices via their corresponding power-vectors. Once the power-vectors, and thus the matrices, are classified, each cluster contains a *medoid* as a representative member. During the simulations (online), the available actions at time step k are identified as the cluster $\mathcal{A}_{\mathcal{G}}{}^{(k)}$, whose medoid achieves the minimum Bhattacharyya distance to $\mathcal{P} = \mathbf{T} \times P_{\mathbf{d}_{k-1}|H_{k-1}}$, where $\mathbf{T}$ is the transition probability matrix associated with Fig. 5. These action restrictions are implemented for the adaptive methods described below.

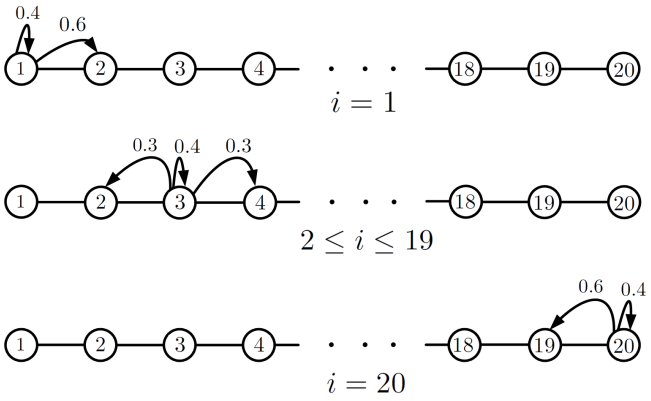


FIGURE 5. State transition law for the moving target.

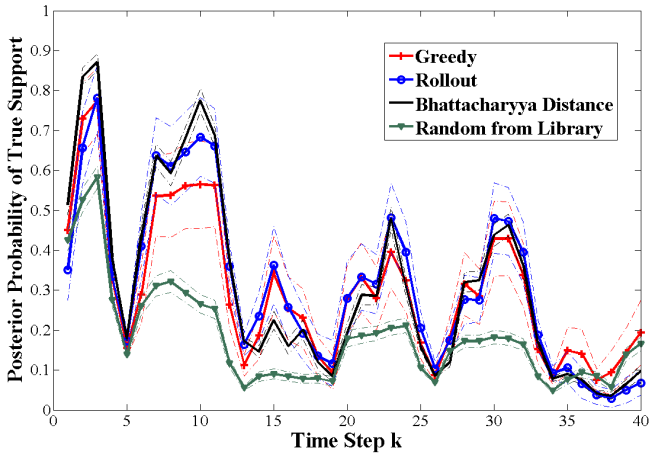


FIGURE 6. Performance comparison of the four policies described in Simulation 2. The dashed lines indicate the 85% confidence intervals.

1. Random from Library: This is the only non-adaptive scheme considered in Simulation 2. It does not abide by the action restrictions of available measurement matrices. Simply, a single measurement matrix is chosen from the library of 4368 matrices at random.

2. Bhattacharyya Distance: This is an adaptive heuristic inspired by the findings in [18] where the measurement should resemble the prior. This is really an extension of the sorting procedure used for clustering measurement matrix library $\mathcal{A}_{\mathcal{G}}{}^{(k)}$. Once the closest medoid of the 400 groups is identified, the Bhattacharyya distance is again used to rank the measurement matrices in the minimum medoid's cluster. The action selected is the measurement matrix with the minimum distance from its power-vector to the predicted support distribution $\mathcal{P}$.

3. Greedy: This action selection process is based *single* step decision-making horizon. Greedy, as with the Bhattacharyya distance, only considers the available actions based on the medoid selection. At each time step k, the best matrix is chosen from the restricted measurement matrix library $\mathcal{A}_{\mathcal{G}}{}^{(k)}$ minimizing the one-step ahead belief cost.

4. Rollout: This solution method selects an action, $u_k \in \mathcal{A}_{\mathcal{G}}{}^{(k)}$ that minimizes the 3-step lookahead Q-value approximation. The base policy used in this approximation is the Bhattacharyya Distance (method 2).

Fig. 6 shows the performance of each method described above over 40 time steps. As expected, the three adaptive methods outperform the non-adaptive method (except in select occlusion areas). However, Greedy and Rollout perform similarly. This simulation shows an example of a case where the multi-step lookahead approach is not gaining extra information compared to the myopic method.

C. SIMULATION 3: 1-SPARSE TIME-VARYING SIGNAL (PROBLEM 2 WITH OCCLUSIONS)

We consider another scenario involving a moving target with occlusions. Here, a single target moves among $N = 40$ possible locations. The transition law that describes the movement of this target is shown in Fig. 5. Like in Simulation 2, we assume a uniform distribution as the prior $P_{\mathbf{d}_0}$, but the occlusions are now located at indices 6 to 10, 16 to 20, and 36 to 40. For simplicity, we consider the case where the target starts in location 2 and moves one location to the right at each time step. This means that over the duration of $m = 10$ time steps, the target will be behind occlusion points from time step 5 to 9. We set γ used in (1) to 300 and we set SNR $= 2.5$ (dB). Finally, in this simulation, we use one signal sample and repeat the experiment 50 times.

We consider this scenario in the context of Problem 2. Here, we adaptively select the number of measurements taken at each time step. We set $l_{\max} = 65$, meaning at most 65 measurements are requested per time step. After the number of measurements l_k is determined, we use a fixed scheme to design the measurement matrix $\mathbf{A}_k$. The following describes the scheme used in this simulation: At each time k the jth entry of the measurement row $\mathbf{A}_k(i)$, i.e., $\mathbf{A}_k(i)(j)$, is generated in the following way:

$$\mathbf{A}_k(i)(j) = \begin{cases} 0, & j \text{ is an occlusion point,} \\ x_{P_{\mathbf{d}_k|H_k}(j|H_k)}, & \text{otherwise,} \end{cases}$$

where the value $x_{P_{\mathbf{d}_k|H_k}(j|H_k)}$ is a sample drawn from the normal distribution $\mathcal{N}(0, P_{\mathbf{d}_k|H_k}(j|H_k))$. Each row vector $\mathbf{A}_k(i)$ is then normalized. The idea of using the function $P_{\mathbf{d}_k|H_k}$ to create row entries in the measurement matrix $\mathbf{A}_k$ comes from [18], one of the early works in adaptive design (1-step) of compressive measurement matrices.

In this simulation, we consider three methods of action selection:

1. Greedy: The best number of measurements which minimizes the 1-step lookahead POMDP belief cost.

2. Smart: A specific number of measurements $l_k \leq l_{\max}$ is selected using the following rule:

$$l_k = \begin{cases} 0, & 0.95 < \alpha, \\ 5, & 0.8 < \alpha \leq 0.95, \\ 15, & 0.65 < \alpha \leq 0.8, \\ 45, & 0.25 < \alpha \leq 0.65, \\ 35, & 0.15 < \alpha \leq 0.25, \\ 25, & \alpha \leq 0.15, \end{cases} \tag{10}$$

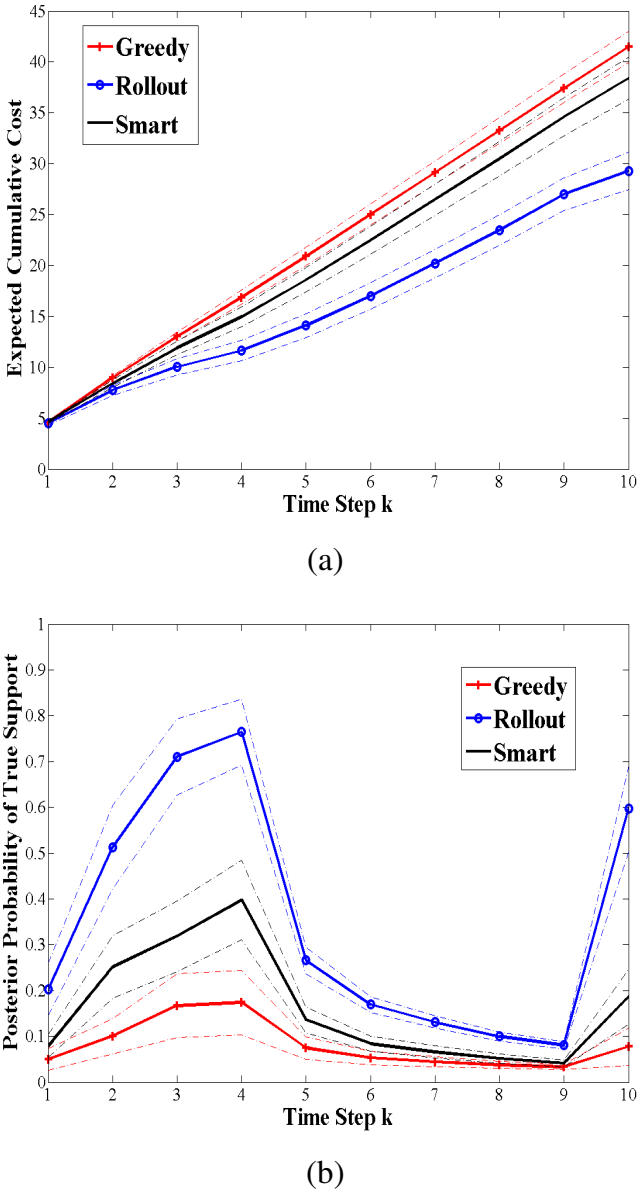


FIGURE 7. Performance comparison of all methods in Simulation 3. The dashed lines indicate the 90% confidence intervals.

where α is the fraction of $P_{\mathbf{d}_k|H_k}$ at time step k that belongs to occlusion points. This policy suggests that when α implies the target is somewhere far from the occlusion points with high probability, a fair (but not exorbitant) number of measurements is used. When the target approaches the occlusion points, the number of measurements is increased to achieve an accurate estimate of the target location before it "disappears." Once the target is occluded, there is no point in making any more measurements until the target becomes visible again.

3. Rollout: This is a 3-step lookahead solution method. Rollout chooses the action that minimizes the total (approximate) cost incurred over three consecutive time steps. For this method, the Smart policy is the base policy of Rollout.

TABLE 1. Target positions over 10 time steps in Simulation 4.

k	1	2	3	4	5	6	7	8	9	10
$\mathbf{d}_k(1)$	2	3	4*	5*	6	7	8*	9	10	11
$\mathbf{d}_k(2)$	18	17	16	15	14	13*	12*	11	10	9

In each run, 150 Rollout trajectories are averaged to estimate the Q-value of each action candidate.

Fig. 7(a) shows the expected cumulative cost of the above three methods. This figure shows that the Rollout method outperforms the Greedy method, and demonstrates the value of multi-step lookahead. Moreover, it is interesting to see that both Greedy and Smart methods perform similarly while the amount of computation required for the Smart policy is much less than that of the Greedy policy.

We have also plotted the posterior probability of true support, i.e., $P_{\mathbf{d}_k|H_k}(\mathbf{d}_T|H_k)$, at each time step k in Fig. 7(b). This figure clearly captures the ability of each method tracking the moving target. When the target is outside the occlusion area, Rollout improves the ability to detect the target location better than the other two methods over time.

D. SIMULATION 4: 2-SPARSE TIME-VARYING SIGNAL (PROBLEM 2 WITH OCCLUSIONS)

Now, we consider a more general setting in which there are multiple moving targets, i.e., a dynamic s-sparse signal. To keep the amount of computation low, we make several simplifying assumptions. First, we only consider $s = 2$ targets that are moving in $\mathbb{R}^{20}$. Note that even for this case, there are $N^s = 400$ possibilities for the support $\mathbf{d}_k$ at each time step. Second, we consider an initial distribution where only 25% of its entries are nonzero with equal values. This prior distribution suggests that the first target is located at one of the locations 1, 2, or 3 and the second target is located either at location 19 or 20. We let $\sigma_w^2 = 1$, and consider the strength values 1.7 and -1.5 for the first and second target, respectively, which means that SNR $= 4$ dB.

Targets 1 and 2 are initially located at positions 1 and 19, and as time evolves they move towards each other one index at a time. The transition law that describes the movement of each of these targets is the same transition law that was used in Simulation 3 and shown in Fig. 5. Note that these targets move independently from each other. Finally, we consider five occlusion points located at positions 4, 5, 8, 12, and 13.

Our simulation runs for 10 time steps. During this time, the two targets temporarily coincide as well as occupy occluded locations. Table 1 summarizes the position of each target at each time step. This table indicates those time steps in which an occlusion occurs by a "$*$" sign next to the position of the target. Based on this table, we expect to see a drop in the performance of all the methods at time steps 3, 4, 6, and in particular 7 (where both targets are occluded). Moreover, at time step 9, when the two targets reach the location 10, the strength value is equal to the sum of the strength values of the two targets. This means that at time $k = 9$, the measurements collected are from a 1-sparse signal with the strength value

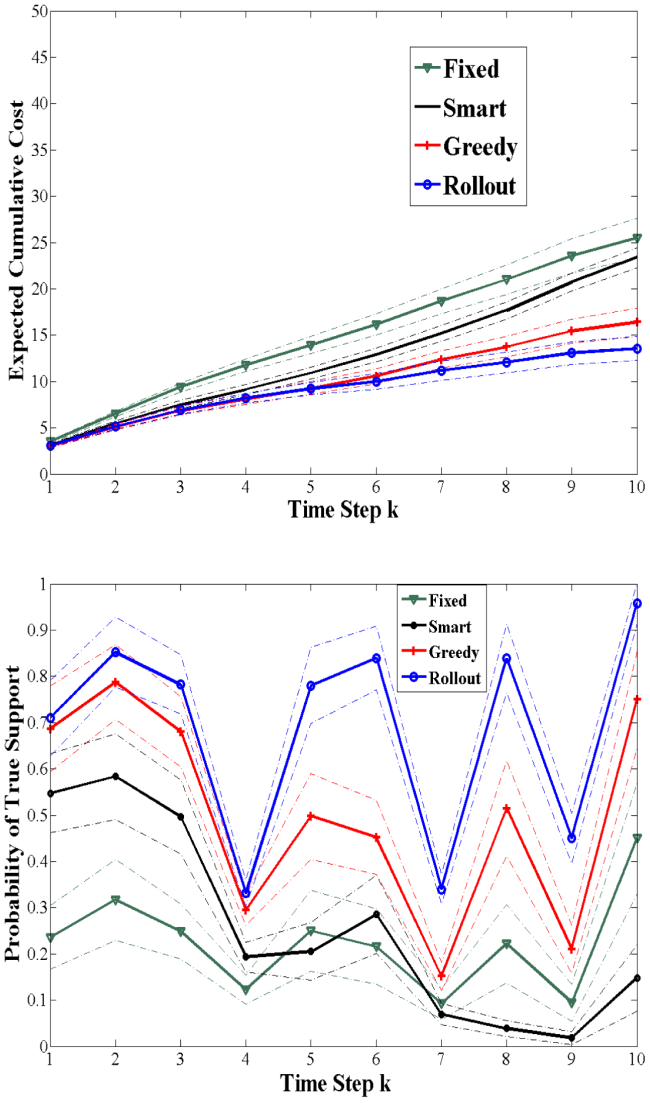


FIGURE 8. Performance comparison of all methods in Simulation 4 when Algorithm 1 is used. The dashed lines indicate the 85% confidence intervals.

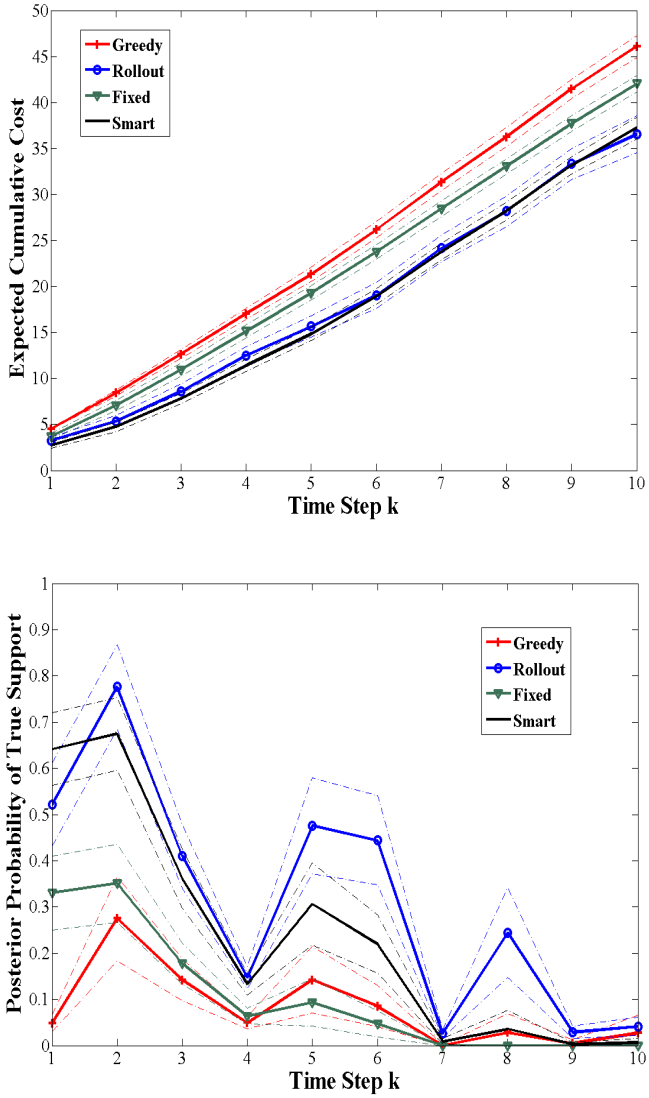


FIGURE 9. Performance comparison of all methods in Simulation 4 when Algorithm 2 is used. The dashed lines indicate the 85% confidence intervals.

$1.7 - 1.5 = 0.2$. Thus, the SNR value drops to -14 dB and dip in performance is expected for all methods.

Similar to the previous simulations, the performance is compared across four approaches in this simulation:

1. Fixed: This is a non-adaptive method, where a fixed value of 30 measurements are taken at each time step.

2. Smart: This method is similar to the Smart method in Simulation 3 but with two differences: a) The method has been modified to work for 2 targets, and b) The number of measurements for the different cases in (10) changes from 0, 5, 15, 45, 35, and 25 to 5, 10, 20, 55, 40, and 30, respectively.

3. Greedy: Similar to the myopic methods in previous simulations, this is the 1-step lookahead solution.

4. Rollout: This method is the 3-step lookahead method with Smart, introduced here, as the base policy.

To estimate the expected cost in both Greedy and Rollout, we generate 25 samples from the belief state at each time step and we repeat the experiment for each trajectory 20 times.

In all of the techniques introduced, once the number of measurements l_k is selected at each time step, we build the measurement matrix the same as in Simulation 3. Moreover, the maximum number of measurements allowed at each time step is $l_{\max} = 65$ and we set γ to be equal to 293. The values shown in the results here are the average values taken over 36 rounds of simulation.

As mentioned in Section II-B, we implement two heuristics, Algorithms 1 and 2, to decrease the amount of computation. In one experiment, Algorithm 1 limits the belief state distribution expansion by maintaining a hypothesis tree with a sliding window of size $w = 4$, resulting in approximately 6000 Kalman filters per time step, after the time step $k = 4$ (and significantly more before this first pruning action). This is clearly more Kalman filter computations than Algorithm 2,

but yields in a higher probability of support identification. In a separate simulation, Algorithm 2 approximates the conditional posterior distribution of the support $P_{\mathbf{d}_k|H_k}$ with a distribution $\hat{P}_{\mathbf{d}_k|H_k}$ that contains only 90 nonzero entries for $k > 1$. Therefore, at any point in our simulation (after the first time step), the belief state update requires only 90 Kalman filters.

Figs. 8 and 9 show the results from running this simulation when Algorithms 1 and 2 are used, respectively. In both set of results, similar to Simulation 3, Rollout has the best performance among all the methods. Also, notice the performance drop of all methods, as expected, in the occlusion and the low SNR areas shown in Table 1.

V. CONCLUSION

In this paper, we studied the problem of adaptively determining compressive measurements for estimating supports of time-varying sparse signals in the context of multi-target tracking. We formulated this problem as a POMDP and used the rollout method to find an approximate solution. We provided several simulations that consider different scenarios for both single moving target (1-sparse) and multiple moving targets (s-sparse). The simulation results indicate that the adaptive techniques have a perform better than the non-adaptive (traditional) designs. Moreover, when comparing the myopic vs. non-myopic variations, the scenario specifications and the POMDP cost play a major role in their relative performance. For example, when we consider occlusion areas in the setting and the goal is to minimize the total number of measurements, the non-myopic design has the best performance. On the contrary, in cases where neither of these conditions hold, there is no difference between the performance of myopic and non-myopic designs and using the latter design only increases the computation cost.

REFERENCES

[1] E. J. Candés and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[2] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[3] E. J. Candés, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, Aug. 2006.

[4] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[5] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*, vol. 95. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[6] R. A. DeVore, "Deterministic constructions of compressed sensing matrices," *J. Complex.*, vol. 23, no. 4, pp. 918–925, Dec. 2007.

[7] P. Indyk, "Explicit constructions for compressed sensing of sparse signals," in *Proc. 19th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2008, pp. 30–33.

[8] W. U. Bajwa, J. D. Haupt, G. M. Raz, S. J. Wright, and R. D. Nowak, "Toeplitz-structured compressed sensing matrices," in *Proc. 14th IEEE/SP Workshop SSP*, Madison, WI, USA, Aug. 2007, pp. 294–298.

[9] X. Weiyu and B. Hassibi, "Compressed sensing over the Grassmann manifold: A unified analytical framework," in *Proc. 46th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Sep. 2008, pp. 562–567.

[10] R. Calderbank, S. Howard, and S. Jafarpour, "Construction of a large class of deterministic sensing matrices that satisfy a statistical isometry property," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 358–374, Apr. 2010.

[11] S. D. Howard, A. R. Calderbank, and S. J. Searle, "A fast reconstruction algorithm for deterministic compressive sensing using second order Reed-Muller codes," in *Proc. 42nd IEEE Annu. CISS*, Princeton, NJ, USA, Mar. 2008, pp. 11–15.

[12] M. A. Davenport, P. T. Boufounos, M. B. Wakin, and R. G. Baraniuk, "Signal processing with compressive measurements," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 445–460, Apr. 2010.

[13] Z. Ben-Haim and Y. C. Eldar, "The Cramér-Rao bound for estimating a sparse parameter vector," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3384–3389, Jun. 2010.

[14] A. Eftekhari, J. Romberg, and M. Wakin, "Matched filtering from limited frequency samples," *IEEE Trans. Inf. Theory*, vol. 59, no. 6, pp. 3475–3496, Jun. 2013.

[15] E. Arias-Castro, E. J. Candès, and M. A. Davenport, "On the fundamental limits of adaptive sensing," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 472–481, Jan. 2013.

[16] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.

[17] E. Bashan, R. Raich, and A. O. Hero, III, "Optimal two-stage search for sparse targets using convex criteria," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5389–5402, Nov. 2008.

[18] R. M. Castro, J. Haupt, R. Nowak, and G. M. Raz, "Finding needles in noisy haystacks," in *Proc. IEEE ICASSP*, Las Vegas, NV, USA, Mar./Apr. 2008, pp. 5133–5136.

[19] D. Wei and A. O. Hero, III, "Multistage adaptive estimation of sparse signals," in *Proc. IEEE Stat. Signal Process. Workshop*, Aug. 2012, pp. 153–156.

[20] J. D. Haupt, R. G. Baraniuk, R. Castro, and R. D. Nowak, "Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements," in *Proc. Conf. Rec. 43rd IEEE Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Nov. 2009, pp. 1551–1555.

[21] D. Sejdinovic, C. Andrieu, and R. Piechocki, "Bayesian sequential compressed sensing in sparse dynamical systems," in *Proc. 48th IEEE Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Sep./Oct. 2010, pp. 1730–1736.

[22] C. Qiu, W. Lu, and N. Vaswani, "Real-time dynamic MR image reconstruction using Kalman filtered compressed sensing," in *Proc. IEEE ICASSP*, Taipei, Taiwan, Apr. 2009, pp. 393–396.

[23] W. Dai, D. Sejdinovic, and O. Milenkovic, "Gaussian dynamic compressive sensing," in *Proc. Int. Conf. SampTA*, Singapore, May 2011, pp. 1–4.

[24] M. S. Asif, D. Reddy, P. T. Boufounos, and A. Veeraraghavan, "Streaming compressive sensing for high-speed periodic videos," in *Proc. 17th IEEE ICIP*, Hong Kong, China, Sep. 2010, pp. 3373–3376.

[25] D. Angelosante, G. B. Giannakis, and E. Grossi, "Compressed sensing of time-varying signals," in *Proc. 16th IEEE Int. Conf. DSP*, Santorini, Greece, Jul. 2009, pp. 1–8.

[26] N. Vaswani, "Kalman filtered compressed sensing," in *Proc. 15th IEEE ICIP*, San Diego, CA, USA, Oct. 2008, pp. 893–896.

[27] L. Weichang and J. C. Preisig, "Estimation and equalization of rapidly varying sparse acoustic communication channels," in *Proc. OCEANS*, Boston, MA, USA, Sep. 2006, pp. 1–6.

[28] M. G. Christensen, J. Stergaard, and S. H. Jensen, "On compressed sensing and its application to speech and audio signals," in *Proc. Conf. Rec. 43rd IEEE Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Nov. 2009, pp. 356–360.

[29] W. Li and J. C. Preisig, "Estimation of rapidly time-varying sparse channels," *IEEE J. Ocean. Eng.*, vol. 32, no. 4, pp. 927–939, Oct. 2007.

[30] M. L. Littman, "A tutorial on partially observable Markov decision processes," *J. Math. Psychol.*, vol. 53, no. 3, pp. 119–125, Jun. 2009.

[31] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, pp. 99–134, 1998.

[32] R. Zahedi, L. W. Krakow, E. K. P. Chong, and A. Pezeshki, "Adaptive compressive sampling using partially observable Markov decision processes," in *Proc. IEEE ICASSP*, Kyoto, Japan, Mar. 2012, pp. 5269–5272.

[33] R. Zahedi, L. W. Krakow, E. K. P. Chong, and A. Pezeshki, "Adaptive compressive measurement design using approximate dynamic programming," in *Proc. ACC*, Washington, DC, USA, Jun. 2013.
[34] J. Vermaak, S. J. Godsill, and P. Perez, "Monte Carlo filtering for multi-target tracking and data association," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 1, pp. 309–332, Jan. 2005.
[35] S. Oh, S. Russell, and S. Sastry, "Markov chain Monte Carlo data association for general multiple-target tracking problems," in *Proc. 43rd IEEE CDC*, vol. 1. Dec. 2004, pp. 735–742.
[36] R. Karlsson and F. Gustafsson, "Monte Carlo data association for multiple target tracking," in *Proc. Target Track., Algorithms Appl.*, vol. 1. Oct. 2001, pp. 13/1–13/5.
[37] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, "Complexity of finite-horizon Markov decision process problems," *J. ACM*, vol. 47, no. 4, pp. 681–720, Jul. 2000.
[38] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, Sep. 2000.
[39] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *Proc. 19th IEEE CDC*, vol. 19. Albuquerque, NM, USA, Dec. 1980, pp. 807–812.
[40] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Syst.*, vol. 29, no. 6, pp. 82–100, Dec. 2009.
[41] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1, pp. 5–18, Jan. 2004.
[42] G. W. Pulford, "Taxonomy of multiple target tracking methods," *IEE Proc. Radar, Sonar Navigat.*, vol. 152, no. 5, pp. 291–304, Oct. 2005.
[43] E. K. P. Chong, C. Kreucher, and A. O. Hero, III, "Partially observable Markov decision process approximations for adaptive sensing," *Discrete Event Dyn. Syst.*, vol. 19, pp. 377–422, Sep. 2009.
[44] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, pp. 99–134, May 1998.
[45] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, Jan. 2007.
[46] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *Eur. J. Control*, vol. 11, nos. 4–5, pp. 310–334, 2005.
[47] J. H. Conway, R. H. Hardin, and N. J. A. Sloane, "Packing lines, planes, etc.: Packings in Grassmannian spaces," *Experim. Math.*, vol. 5, no. 2, pp. 139–159, Apr. 1996.
[48] R. Zahedi, A. Pezeshki, and E. K. P. Chong, "Measurement design for detecting sparse signals," *Phys. Commun.*, vol. 5, no. 2, pp. 64–75, Jun. 2012.
[49] H. Park and C. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009.

RAMIN ZAHEDI received the B.Sc. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 2003, and the M.Sc. degree in electrical and computer engineering from Colorado State University, Fort Collins, CO, USA, in 2007, where he is currently pursuing the Ph.D. degree in electrical engineering. His current research interests include signal processing, dynamic programming and adaptive control, compressive sensing, and optimization.

LUCAS W. KRAKOW received the B.S. (Hons.) degree in applied mathematics from Chadron State College, Chadron, NE, USA, in 2002, and the M.S. degree in computational mathematics from Colorado State University, Fort Collins, CO, USA, in 2009. He has participated in an electrical and computer engineering research group focused on target tracking with unmanned aerial vehicles from 2006 to 2008 with the U.S. Air Force Academy, Colorado Springs, CO, USA. He is pursuing the Ph.D. degree in electrical and computer engineering with Colorado State University, where he is a Graduate Research Assistant. He engages in external research opportunities as a Private Contractor. His current research interests include target tracking, sensor network management, and controls optimization.

EDWIN K. P. CHONG (F'04) received the B.E. (Hons.) degree from the University of Adelaide, Adelaide, Australia, in 1987, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 1989 and 1991, respectively, where he held an IBM Fellowship. He joined the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 1991. Since August 2001, he has been a Professor of electrical and computer engineering and a Professor of mathematics with Colorado State University, Fort Collins, CO, USA. He has co-authored the best-selling book *An Introduction to Optimization* (4th Ed., Wiley-Interscience, 2013). He is currently a Senior Editor of the IEEE Transactions on Automatic Control and an Editor for *Computer Networks* and the *Journal of Control Science and Engineering*. He served as the IEEE Control Systems Society Distinguished Lecturer. He received the National Science Foundation CAREER Award in 1995 and the ASEE Frederick Emmons Terman Award in 1998. He was a co-recipient of the 2004 Best Paper Award for a paper in the journal computer networks. In 2010, he received the IEEE Control Systems Society Distinguished Member Award. He currently serves as a Vice President for Financial Activities in the IEEE Control Systems Society.

ALI PEZESHKI (S'95–M'05) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 1999 and 2001, respectively, and the Ph.D. degree in electrical engineering from Colorado State University, Fort Collins, CO, USA, in 2004. In 2005, he was a Post-Doctoral Research Associate with the Electrical and Computer Engineering Department, Colorado State University. From 2006 to 2008, he was a Post-Doctoral Research Associate with the Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ, USA. Since August 2008, he has been an Assistant Professor with the Department of Electrical and Computer Engineering, Colorado State University. Since October 2012, he has been an Assistant Professor of mathematics with Colorado State University. His current research interests include statistical signal processing and coding theory and their applications to distributed sensing, active/passive sensing, and bioimaging. He is an Editorial Board Member of the IEEE Access.