Using Diffusive Protocols for Distributed MAP Decision-Making and HMM Estimation

Sandip Roy * Ali Saberi *

January 18, 2005

Abstract

We explore the use of diffusive agreement protocols in addressing the problem of global sensor fusion. In particular, we show that diffusive (averaging-based) protocols can be applied to achieve distributed maximum \textit{a posteriori} (MAP) detection or decision-making in a network of sensors. That is, we consider a network of sensors that make noisy measurements of an underlying discrete-valued scene, and pursue distributed computation of the MAP estimate for this underlying scene given the observations throughout the network. We show that use of a diffusive (averaging-based) agreement protocol permits computation of this MAP estimate by all the sensors, under broad conditions on the communication topology of the network. Finally, we enhance our algorithm for MAP detection to achieve distributed estimation for a Hidden Markov Model (HMM).

*Both authors are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, U.S.A., \{sroy,saberi\}@eecs.wsu.edu. The second author was supported by by the Office of Naval Research under grant number N000140310848.
1 Introduction

Diffusive protocols for consensus and agreement have received considerable attention in the controls literature recently, as tools for distributed decision-making and/or stabilization in such application areas as unmanned aerial vehicle (UAV) control and sensor network operation (e.g., [1, 2, 3, 4, 5, 6, 7, 8, 9]). The article [6] exposes that a distributed diffusive consensus protocol can be used to fuse noisy observations made by a network of sensors, in such a way that each agent eventually obtains the maximum likelihood (ML) estimate of a continuous-valued underlying state. A complementary task to the estimation task pursued in [6] is that of decision-making or detection: agents in a sensor network are often required to decide among a finite set of options (e.g., the presence or absence of a target, or the warning status associated with a chemical spill), using corrupted observations of these underlying options. In this article, we show that an agreement protocol can be used to achieve maximum a posteriori probability (MAP) detection or decision-making in a totally distributed manner. Further, we use diffusive protocols to achieve totally distributed detection of a Markovian state from a sequence of observations—i.e., for Hidden Markov Model (HMM) filtering.

The remainder of the article is organized as follows: in section 2, we summarize the use of diffusive protocols for agreement, in the process reviewing some of the literature in this field. We also discuss, in general terms, the application of diffusive agreement protocols for estimation and detection, with the aim of exposing key tasks required for distributed estimation and detection. In Section 3, we precisely formulate the problem of distributed MAP detection, and show that a diffusive agreement protocol can be used to solve this problem. In Section 4, a distributed protocol for HMM filtering that recursively applies the distributed MAP detection protocol is developed. A canonical example is used throughout to illustrate our development.
2 Review: Agreement and its Application to Estimation/Detection

In several fields, networks of agents are required to cooperatively complete a task, e.g. a swarm of military vehicles must find and destroy a target or a network of sensors must detect a heart arrhythmia and alert the patient. In completing such a dynamic task, it is often necessary for a network to come to an agreement on a topic, e.g., whether or not the target is present or what the patient’s heart rate is. Hence, the problem of agreement among a network of agents is of importance. Protocols for agreement (rules using which a network can reach agreement) have been studied in the computer science community for several years (see [10] for a thorough introduction). Recently, there has been interest in the control community on diffusive protocols for agreement, i.e. ones in which agents recursively update their opinions about the the topic of interest through linear averaging, until an agreement is reached (see, e.g., [6, 7, 4, 5, 8, 9, 11]). This strategy of reaching agreement through linear averaging is a compelling one for engineering applications, both because averaging is a natural computation for some mechanical and electrical devices and because the use of averaging permits significant analysis using linear system and control theory. The tractability of diffusive protocols is especially appealing in that protocol performance can be tied to the communication topology of network (e.g., [4, 8, 7]), thus giving significant insight into the design of the topology. Further, the use of diffusive protocols makes possible the analysis of several realistic aspects in sensor network applications, in particular, such as the likelihood of faults and the impact of asynchronicity in communication [9, 6].

Let us now review a mathematical formulation for diffusive agreement protocols (see, e.g., [4, 8]), and in the process present the notation that is used throughout the remainder of the paper. We consider a network of of \( n \) agents, labeled \( 1, \ldots, n \). Each agent \( i \) has an initial opinion \( x_i[0] \in R \) about a topic of interest\(^1\). Also, we assume that each agent \( i \) has access (through communication) to the current opinions of a set of neighboring agents \( \mathcal{N}(i) \), including possibly the agent \( i \) itself. To illustrate this communication topology, we find it convenient to define a

\(^1\)For simplicity, we assume that agents have scalar observations, though the generalization to the vector case is straightforward.
network graph, which comprises $n$ vertices and a set of directed edges. In particular, a directed edge is drawn from vertex $j$ to vertex $i$ if and only if $j$ is a neighbor of $i$, i.e $j \in \mathcal{N}(i)$. We shall often consider one broad class of network graphs: ergodic graphs are those in which, for any sufficiently large integer $L$, there is a path of $L$ steps between each pair of sensors (see [12] for a detailed introduction to ergodicity). An ergodic network graph comprising 5 sensors is shown in Figure 1.

![Network Graph](image)

Figure 1: The network graph for a network with five sensors is shown. The graph is ergodic.

With the motivation of reaching agreement, the agents recursively update their opinions based on their neighbors, according to a linear rule:

$$ x_i[k+1] = \sum_{j \in \mathcal{N}(i)} d_{ij} x_j[k]. \quad (1) $$

By stacking the opinions into a single vector $\mathbf{x}[k] \triangleq \begin{bmatrix} x_1[k] \\ \vdots \\ x_n[k] \end{bmatrix}$ and aggregating the weights $d_{ij}$ into a protocol matrix $D \triangleq [d_{ij}]$, we can write the recursions for the agents’ opinions concisely as

$$ \mathbf{x}[k+1] = D\mathbf{x}[k]. \quad (2) $$
We shall refer to Equation 2 as a **linear protocol**. If the opinions of all the agents converge to the same value regardless of the initial opinions of the agents when a particular linear protocol is used, we shall call that protocol a **linear agreement protocol** or simply and **agreement protocol**.

We shall refer to the common asymptotic opinion shared by the agents as the **agreement value**. It is easy to check (see, e.g., [7, 8]) that a protocol is a linear agreement protocol if and only if the eigenvalues of \( D \) are less than or equal to 1 in magnitude, and any eigenvalue of \( D \) at 1 is strictly dominant and has associated right eigenvector \( \mathbf{1} \). In general, the agreement value is a linear function of the initial opinions of the agents, i.e. \( \lim_{k \to \infty} x_i[k] = p' \mathbf{x}[0] \), for some \( p' \). We shall refer to this dependence of the agreement value on the initial opinions of the agents—or, equivalently, to the vector \( p' \)—as the **agreement law** for the network\(^2\). It is easy to check that the agreement law \( p' \) is given by the left eigenvector of \( D \) associated with the unity eigenvalue if \( D \) indeed has an eigenvalue at 1, and is the zero vector otherwise. Since the key features of an agreement protocol’s dynamics are summarized by its initial condition vector, protocol matrix, and agreement law, we shall often refer to an agreement protocol by the triple \( (\mathbf{x}[0], D; p') \).

This article is concerned with the use of distributed agreement protocols in a specific application, in particular MAP detection. Since we plan to design an agreement protocol—or in other words the protocol matrix \( D \)—to achieve the desired detection task, it is helpful to specify conditions on a network’s topology given which protocol design is possible. A variety of graphical conditions that permit development of agreement and/or average consensus protocols have been developed (e.g., [4, 8]). A relevant condition is presented in the following theorem, which was

---

\(^2\)The notion of an agreement law was first introduced in our earlier work [8]. The other works on diffusive protocols note the possibility for different dependences of the agreed-upon value on the initial conditions, but typically enforce that the agreement value is a fair arithmetic average (referred to as **average consensus**) of the initial opinions of the agents. In other words, they study the case where the agreement law is \( p' = \left[ \frac{1}{n} \ldots \frac{1}{n} \right] \). When such protocols are used, arbitrary dependence of the agreed-upon value on the initial conditions can still be achieved through appropriate scaling of the initial and final opinions of the agents. However, allowing the freedom of arbitrary agreement laws has several advantages, including permitting agreement in networks in which average consensus is impossible (see [8]) and permitting faster agreement (a topic we will discuss in a future paper). For this reason, we explicitly consider agreement laws here.
developed in our earlier paper [9]:

**Theorem 1** For a network with an ergodic graph, we can design an agreement protocol that achieves any desired agreement law $\mathbf{p}'$ such that $\mathbf{p} > 0$ and $\mathbf{p}' \mathbf{1} = 0$. That is, we can construct a protocol matrix to achieve agreement and obtain any desired positive linear agreement law (to within a scale factor)$^3$.

We refer the reader to [8] for details on the construction of the protocol matrix.

Motivated by the need for robustness to communication failures, a couple recent works (e.g., [11, 6]) have developed agreement protocols for networks with time-varying topologies. In [6], a time-varying protocol is shown to achieve agreement, given that the network graph is symmetric (undirected) and that average consensus (i.e., agreement law $\mathbf{p}' = \left[ \frac{1}{n}, \ldots, \frac{1}{n} \right]$) is desired. We shall discuss the implication of this result in the context of our detection algorithm later in this article.

Our agreement protocol for MAP detection is closely connected with the protocol for ML estimation developed in [6], and hence we find it useful to review [6] to identify features required of our protocol. The article [6] seeks a distributed protocol using which a network of sensors can estimate an unknown, non-random continuous-valued scene (state in their terminology). Each sensor makes a measurement of the underlying scene; the measurement is assumed to be a linear function of the scene corrupted by additive Gaussian noise. The noise at each sensor is assumed independent.

The article [6] uses two interwoven agreement protocols to achieve **distributed ML estimation** (i.e., ML estimation of the scene by each sensor in the network, using distributed communication). Although the two agreement protocol updates for distributed estimation are considered together in [6], we view these two updates as performing different tasks:

- The **basic task**—achieved by one of the interwoven protocols—is to distributedly combine the observations to obtain the ML estimate, assuming that each agent has sufficient knowledge of the global statistics of the noise and the network topology to do so. In many applica-

$^3$The notation $\mathbf{p} > 0$ indicates that the vector $\mathbf{p}$ is elementwise positive.
tions, we believe it reasonable that this statistical and structural information would be built into the sensor design or obtained periodically upon change in the network topology, rather than obtained continually by the sensors through measurement. Especially in applications where sensors are very simple devices, it is unlikely that these sensors could measure their noise statistics and would instead depend on hard-coded information about the noise to determine their update strategy. In these cases, computing the ML estimate from the observations is the only required distributed task.

- A secondary task achieved in [6] is the distributed computation of the parameters of the network, or more precisely the computation of the statistical and topological information needed by each agent to achieve the basic task. Specifically, the article [6] shows that a statistic that is needed to compute the ML estimate (in a terminal scaling step) can itself be found using a distributed agreement protocol. Also, the article suggests use of a protocol matrix containing the Metropolis-Hastings weights, for which the parameters needed by each agent can be determined through local communication (in particular, through identification of the number of neighbors of that agent). We believe that this further computation of statistical and topological information is especially important for networks that are commonly subject to changes in topology.

3 A Diffusive Protocol for MAP Detection

In this section, we address the question of distributed detection of a discrete-valued underlying scene from a network of noisy sensor measurements of that scene. In particular, we develop an algorithm using which each sensor in the network can compute the most probable underlying scene, given the observations of all the sensors. That is, we show how the global MAP estimate for the underlying scene can be found in a distributed manner. Our strategy is based on applying a diffusive agreement protocol. We first describe the network model and detection problem, and then present and discuss the algorithm.
3.1 The Model

We consider a network of $n$ sensors (agents), that are tasked with detecting (deciding on) an underlying discrete-valued scene (option), which we represent using the discrete-valued random variable $s$. The random variable $s$ is assumed to take on $m$ outcomes, labeled $1, \ldots, m$. We use the notation $P(s)$ for the vector of probabilities

$$
P(s) = \begin{bmatrix} P(s = 1) \\
\vdots \\
P(s = m) \end{bmatrix}.
$$

Each sensor makes a discrete-valued observation of the underlying scene. The observation $\Omega_i$ made by each sensor $i \in 1, \ldots, n$ is assumed to be a discrete-valued random variable that takes on $m_i$ outcomes, labeled $1, \ldots, m_i$. We assume that the observation made by each sensor $i$ is dependent on the underlying scene but independent of the other observations; in particular, we use the notation $P(\Omega = l | s = j)$ for the conditional probability that the observation made by sensor $i$ is $l$, given that the underlying scene is $j$. We find it convenient to stack the probabilities that an agent makes a particular observation given each underlying scene into a single vector, i.e. as

$$
P(\Omega_i = l | s) = \begin{bmatrix} P(\Omega_i = l | s = 1) \\
\vdots \\
P(\Omega_i = l | s = m) \end{bmatrix}.
$$

Our aim is to develop a totally distributed protocol using which each sensor can achieve MAP detection of the underlying scene. We assume that each sensor $i$ has knowledge of its own observation, say $\Omega_i = o_i$ in a particular experiment. Also, we assume that the sensors can communicate according to an ergodic network graph. Assuming an ergodic network topology is reasonable, since we would expect that each sensor could communicate with all other sensors in several steps, and further we would expect that sensors could access their own opinions. Further, we assume that one sensor—say Sensor 1, w.l.o.g.—has knowledge of the prior probabilities of of each outcome, or in other words of $P(s)$.

Our aim is to develop a distributed algorithm using which each sensor can determine the MAP estimate of the underlying scene given the observations of all the sensors, i.e. $\hat{s}_{MAP} = \arg \max_{j=1,\ldots,m} P(s = j | \Omega_1 = o_1, \ldots, \Omega_n = o_n)$. We will use a diffusive agreement protocol (in
particular, an average consensus protocol) as the key step in an algorithm for MAP detection. This algorithm is developed and characterized in the next subsection.

3.2 The Algorithm

In order to apply a diffusive protocol in achieving distributed MAP detection, it is first helpful to rephrase the MAP detection problem. Notice that we need for our algorithm to generate the estimate

$$\hat{s}_{\text{MAP}} = \arg \max_{j=1,\ldots,m} P(s = j \mid \Theta_1 = \theta_1, \ldots, \Theta_n = \theta_n).$$

However, based on standard optimization concepts, the estimate can equivalently be found as

$$\hat{s}_{\text{MAP}} = \arg \max_{j=1,\ldots,m} \ln[P(s = j \mid \Theta_1 = \theta_1, \ldots, \Theta_n = \theta_n)].$$

Using the definition of condition probability and noting that the marginal probability of the $n$ observations does not depend on the optimization parameter, we can further rewrite $\hat{s}_{\text{MAP}}$ as

$$\hat{s}_{\text{MAP}} = \arg \max_{j=1,\ldots,m} \sum_{i=1}^{n} \ln[P(\Theta_i = \theta_i \mid s = j)] + \ln[P(s = j)].$$

Thus, trivially, $\hat{s}_{\text{MAP}}$ can alternately be viewed as the maximum entry of the vector

$$\mathbf{f} \triangleq \frac{1}{n} \sum_{i=1}^{n} (\ln[P(\Theta_i = \theta_i \mid s)] + \ln[P(s)]),$$

where the natural logarithm of a vector is a component-wise operation, and where we have put in the scaling factor $\frac{1}{n}$ to make clear the applicability of an agreement protocol in computing $\mathbf{f}$. We refer to $\mathbf{f}$ as the log a posteriori probability vector.

We now specify an algorithm using which each sensor can compute $\mathbf{f}$, and hence can determine the MAP estimate $\hat{s}_{\text{MAP}}$. The algorithm comprises the following three steps:

- Each sensor $i$, $2 \leq i \leq n$, computes its initial log probability vector
  $$\mathbf{r}_i[0] = \ln[P(\Theta_i = \theta_i \mid s)].$$
  Meanwhile, sensor 1 computes its initial log probability vector as $\mathbf{r}_1[0] = \ln[P(\Theta_i = \theta_i \mid s)] + \ln[P(s)]$. These vectors serve as the initial opinions for a diffusive agreement protocol.
An agreement protocol with agreement law \( p' = \left[ \frac{1}{n} \ldots \frac{1}{n} \right] \) (i.e., an average consensus protocol) and initial opinions \( r_i[0] \) is applied to the network. (To be precise, since we defined agreement protocols as operating on scalar initial opinions, this computation is actually a repeated application of the same agreement protocol for each entry of the initial log probability vectors.) Noting that \( \overline{r} = \frac{1}{n} \sum_{i=1}^{n} r_i[0] \), we see that every sensor determines \( \overline{r} \) asymptotically using this protocol. Since our network topology is assumed to be ergodic, we see from Theorem 1 that an agreement protocol can indeed be developed to achieve the desired agreement law and hence to distributedly compute \( \overline{r} \).

Each sensor locally determines the global MAP estimate for the underlying scene by choosing the maximum entry of the vector \( \overline{r} \).

Hence, we have developed an algorithm using which each sensor can compute the global MAP estimate for the underlying scene.

There are several points worth discussing about the MAP detection algorithm:

1) Since the logarithm of 0 is undefined, the algorithm needs refinement for the case where prior probabilities for the scene or conditional probabilities for the observations given the scene are zero. One approach for refining the algorithm is to use a special symbol for the logarithm of 0, and to indicate the sum of this symbol with other logarithmic quantities with the symbol itself. If the use of a special symbol is beyond the functionality of the network, another possibility is to replace the zero probabilities by vanishingly small ones. If these probabilities are chosen to be sufficiently small, it can be shown that the limiting vector of the agreement protocol is close to the desired vector \( \overline{r} \).

2) Our algorithm for distributed MAP detection is quite similar in structure to the algorithm for distributed ML estimation discussed in [6], but a couple differences are worth noting. Of course, in order to use a diffusive protocol for detection, we require computing probabilities and taking their logarithms, in contrast to the protocol for ML estimation. A second difference between the algorithms is that our detection algorithm can generate the exact result in
finite time. That is, even though \( \mathbf{f} \) is computed exactly only asymptotically, the estimate for the scene can be generated in finite time because only the proper ordering of the values in \( \mathbf{f} \) is needed to find the MAP estimate. A third difference between the two algorithms is that our detection algorithm does not require scaling of the final value generated by the agreement protocol. As discussed below, this is of importance when considering the secondary task of providing each agent with the required network parameters.

3) We have shown how the basic task of MAP detection can be achieved; some discussion of the secondary task of providing sensors with needed parameters is helpful at this point. In analog with the estimation task discussed in [6], detection in a sensor network may first require that parameters for the algorithm be computed in a distributed manner. Specifically, it may in general be necessary to identify in a distributed manner the a priori probabilities for the scene, the weights in the protocol matrix, and any other scaling factors used in the algorithm. In fact, if knowledge of the priors is distributed, one can easily modify the algorithm so that each sensor incorporates a subset of the priors in computing \( r_i \). Unlike [6], our algorithm does not require scaling the agreement protocol result by a global parameter, and hence no distributed computation of such a parameter is needed. Thus, only distributed computation of the protocol matrix is needed. In the case where the network graph is symmetric, a set of protocol weights known as the Metropolis-Hastings weights can be determined locally by the sensors, as described in [6]. As we have mentioned before, we expect to address the performance of protocols with asymmetric agreement laws (i.e., non-average consensus laws) and/or asymmetric network graph in a future article; we relegate discussion of the distributed computation of the protocol matrix to there. As before, we stress that distributed computation of the parameters (including the protocol matrix) may be unnecessary in many applications, since this information can be hard-coded into the sensors.

4) Our modeling framework can be tractably generalized in a couple respects. First, if the network graph is symmetric but time-varying, MAP detection can still be achieved by modifying our algorithm. In particular, a time-varying agreement protocol can be used to achieve
the detection task. We refer the reader to [6] for details about the time-varying protocol. A
second generalization that is tractable is detection given vector observations for each sensor,
or of a vector scene. Vector observations can be dealt with simply by modifying the calcu-
lation of $r_i[0]$ for each sensor. Meanwhile, detection of a vector scene can be achieved by
representing enumerating the values taken on by the vector as a scalar.

5) A worthwhile generalization for both our study and the distributed estimation study of [6] is
to allow for correlation among the observations made by different sensors. Such correlated
observations are likely to occur in sensor networks due to, e.g., environmental noise that
impacts a group of sensors. The estimation problem posed in [6] can be generalized to allow
correlation by modeling observations made by multiple agents as jointly Gaussian, rather
than as independent Gaussian random variables. Correlations among observations in our
detection problem can, e.g., be modeled using Markov Random Field models [13]. For both
the estimation and detection problem, it is easy to check that an agreement protocol can be
applied to achieve the basic task even when the sensor observations are correlated. What
is more difficult when parameters are correlated is to achieve the secondary task, i.e. to
provide sensors with the necessary parameters for the estimation or detection algorithms. In
the correlated case, the parameters required by each sensor depend in a complicated way on
the local statistics and cross-statistics of the sensors’ observations. We leave for future work
the distributed computation of these parameters, for the case where the parameters cannot be
pre-computed and hard-coded.

6) It is worth noting that we have also studied the distributed decision-making (detection) prob-
lem in our earlier works [8, 9]. In those articles, we sought to achieve distributed decision-
making using a quasi-linear stochastic automaton known as the influence model. In contrast
to our current development, the approach pursued in [8, 9] does not generate a MAP estimate,
but does have the advantage of robustness to parameter uncertainties and certain heteroge-
nieties among the sensors. We leave it to future work to explore further the advantages and
disadvantages of each approach.
Table 1: The estimate of the scene made by each sensor at each step of the diffusive protocol is shown.

### Example

For illustration, we have applied the distributed MAP detection algorithm to a canonical example with five sensors. The sensors can communicate according to the network graph shown in Figure 1. In this example, the underlying scene can take on three values, say *Hot* (*H*), *Warm* (*W*), and *Cold* (*C*), with a priori probabilities $\frac{1}{4}$, $\frac{1}{2}$, and $\frac{1}{4}$, respectively. The five sensors make noisy measurements of this underlying scene. In particular, each sensor makes an observation of *H*, *W*, or *C*, according to the probability table $\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.6 & 0.2 \end{bmatrix}$. For instance, if the underlying scene is *H*, each sensor observes *H*, *W*, and *C*, with probabilities 0.7, 0.2, and 0.1, respectively.

The agents are tasked with estimating the underlying scene from their observations. We apply the developed algorithm to complete this decision task. Here, we illustrate the operation of the algorithm in a particular experiment. Specifically, say that the five sensors have made the following
measurements: Sensor 1 measures $W$, Sensor 2 measures $C$, and Sensors 3, 4, and 5 measure $H$. It is easy to check that the MAP estimate for the underlying scene is $H$. This MAP estimate is indeed distributedly computed by all five sensors using our algorithm, after 8 steps of the diffusive protocol. The estimate of each sensor at each step of the diffusive protocol is shown in Table 1. Also, the log-probability output for Sensor 1 at each step of the diffusive protocol (from which the MAP estimate is selected) is shown in Figure 2. Notice that the log-probabilities require about 20 steps to settle to their final values, but MAP detection is achieved within 8 steps nevertheless.

![Agreement Protocol Results for Sensor 1](image)

**Figure 2:** The log-probability computation for Sensor 1 at each step of the diffusive protocol (from which the MAP estimate is selected). We note that MAP detection (of the underlying scene $H$) is achieved in 8 steps.

### 4 A Diffusive Viterbi Algorithm for Distributed Hidden Markov Models

We have so far shown how a MAP estimate of a discrete-valued underlying scene can be found from distributed discrete-valued observations using a diffusive protocol. A diffusive strategy can
in fact also be used to achieve MAP estimation of a discrete-state Markov process from distributed, discrete-valued observation sequences. In particular, we consider a network of sensors (i.e., a set of sensors that are constrained to communicate along the branches of a network graph) that observe a hidden Markov chain. The problem of MAP estimation of the state sequence from the observation sequence in this model is the classical Hidden Markov Model (HMM) filtering problem, which is solved in a centralized context using the Viterbi algorithm (e.g., [14]). Here, we show that each stage of the Viterbi algorithm can be achieved in our distributed model by applying a diffusive agreement protocol. That is, we present a diffusive algorithm using which each sensor can obtain the MAP state sequence estimate at each stage of the Viterbi algorithm—as well as the conditional MAP estimates required for the next stage. Using the diffusive algorithm at each Viterbi stage, we are thus able to obtain the MAP state sequence estimate.

4.1 The Model

We consider a network of \( n \) sensors. These sensors are tasked with detecting the state sequence \( \{s[k]\} \) of an underlying \( m \)-state hidden Markov chain. We denote the transition matrix and initial probabilities for this hidden Markov chain by \( P = [p_{ij}] \) and \( \pi_0 \), respectively. We assume that at least one sensor in the network—say (w.l.o.g.) Sensor 1—has available the vector these two network parameters.

Each sensor \( i \) makes a sequence of discrete-valued observations. The observation made by agent \( i \) at time \( k \) is assumed to be in the set \( 1, \ldots, m_i \). We denote the observation made by sensor \( i \) at time \( k \) by \( O_i[k] \), and use the notation \( O_i[k_1 : k_2] \) for the sequence of observations over the interval \( [k_1, k_2] \). Each observation \( O_i[k] \) is assumed to be independent of all past dynamics of the model and of the current observations of the other sensors, given the current state of the hidden Markov chain. These conditional probabilities (for \( O_i[k] \) given \( s[k] \)) are parameters of the network, and are assumed known by agent \( i \). Also, we find it convenient to consider the observations made by the sensors together, and hence we define the notation \( O[k] \) for the set of observations \( O_1[k], \ldots, O_n[k] \). Similarly, we use the notation \( O[k_1 : k_2] \) for the sequence \( O[k_1], \ldots, O[k_2] \).
Our aim is to develop a totally distributed protocol using which each sensor can achieve global filtering, i.e. MAP detection of the underlying state sequence \( s[0], \ldots, s[k] \) from the observation sequence \( \mathcal{O}[0 : k] \). Specifically, we assume that each sensor \( i \) has knowledge of its own observation sequence, say \( \mathcal{O}_i[1 : k] = o_i[1 : k] \) in a particular experiment. Also, we assume that the sensors can communicate according to a network graph that is ergodic.

We have thus specified a discrete-valued stochastic process model with distributed observation topology. We shall refer to this model as a \textit{distributed hidden Markov model} (DiHMM).

### 4.2 A Distributed Viterbi Algorithm

Our aim is to develop a distributed algorithm for MAP detection of the underlying state sequence of a DiHMM from the observation sequence. That is, we wish to develop a totally distributed algorithm using which each sensor can find

\[
\hat{s}[0], \ldots, \hat{s}[k] = \arg \max_{j_0 \in [1,m], \ldots, j_k \in [1,m]} \mathcal{P}(s[0] = j_0, \ldots, s[k] = j_k | \mathcal{O}[1 : k]). \tag{5}
\]

It is well known that this optimal state sequence can be found in a centralized manner using a dynamic programming strategy known as the \textit{Viterbi algorithm}. Here, we briefly review the Viterbi algorithm, and then show that each stage of the algorithm can be achieved in a totally distributed manner, using the diffusive algorithm developed in Section 3.

Briefly, the Viterbi algorithm can be constructed by rewriting Equation 5, and hence noting that \( \hat{s}[k] \) can be found as

\[
\hat{s}[k] = \arg \max_{j_k \in [1,m]} \max_{j_k-1 \in [1,m]} \max_{j_k-2 \in [1,m]} \ldots \max_{j_1 \in [1,m]} \ln(\mathcal{P}(s[0] = j_0, \ldots, s[k] = j_k | \mathcal{O}[1 : k]))
\]

\[
= \arg \max_{j_k \in [1,m]} \max_{j_k-1 \in [1,m]} \ldots \max_{j_1 \in [1,m]} \ln(\mathcal{P}(s[k] = j_k, \mathcal{O}[k] | s[k-1] = j_{k-1}, \mathcal{O}[k-1]))
\]

\[
+ \max_{j_k \in [1,m]} \max_{j_k-1 \in [1,m]} \ldots \max_{j_1 \in [1,m]} \ln(\mathcal{P}(s[0] = j_0, \ldots, s[k-1] = j_{k-1}, \mathcal{O}[1 : k-1]))) \tag{6}
\]

From this form, we see that, if the variables

\[
\alpha_{k-1}(j) \triangleq \max_{j_1 \in [1,m], j_{k-2} \in [1,m]} \ln(\mathcal{P}(s[0] = j_0, \ldots, s[k-1] = j, \mathcal{O}[1 : k-1]))
\]

is known for each \( j \in [1, m] \), then the corresponding variables at the next stage
\[ \alpha_k(j) = (\max_{j_1 \in [1,m], \ldots, j_{k-1} \in [1,m]} \ln(P(s[0] = j_0, \ldots, s[k] = j, \Theta[1:k])), \]
can be computed for each \( j \in [1,m] \). In particular, the variables at the next stage are given by
\[ \alpha_k(j) = \max_{j_{k-1} \in [1,m]}[\ln(P(s[k] = j, \Theta[k] | s[k-1] = j_{k-1}, \Theta[k-1]) + \alpha_{k-1}(j_{k-1})]. \]
Also, one can easily check that the optimal state sequence such that the state is \( j \) at time \( k \) contains the optimal state sequence with time \( k-1 \) state given by
\[ \arg \max_{j_{k-1} \in [1,m]}[\ln(P(s[k] = j, \Theta[k] | s[k-1] = j_{k-1}, \Theta[k-1]) + \alpha_{k-1}(j_{k-1})]. \]
We can then find \( \hat{s}[k] = \arg \max_{j} \alpha_k(j) \) and in turn the optimal state sequence \( \hat{s}[0], \ldots, \hat{s}[k] \). We note that the recursive algorithm is initialized using \( \alpha_0(j) = \pi_0(j) \), i.e. by the \( j \)th element of the a priori probability vector \( \pi \). Thus, we have specified the Viterbi algorithm; we refer the reader to [14] for further details.

Notice that stage \( k \) of the Viterbi algorithm requires calculation of
\[ \alpha_k(j) = \max_{j_{k-1} \in [1,m]}[\ln(P(s[k] = j, \Theta[k] | s[k-1] = j_{k-1}) + \alpha_{k-1}(j_{k-1})]. \]
(In turn, the optimal state sequence corresponding to each possible time-\( k \) stage can be found.) Now, we show that calculation of the \( \alpha_k(j) \) can in fact be achieved in a distributed manner. In particular, let us assume that each sensor has available \( \alpha_{k-1}(j), j \in 1, \ldots, m \), and seeks to compute \( \alpha_k(j) \) using the observations \( O[k] \) made by all the sensors. We notice that the formula for \( \alpha_k(j) \) can be expressed in terms of the observations made by the different sensors by exploiting the conditional independence of the observations:
\[ \alpha_k(j) = \max_{i \in [1,m]} \left[ \sum_{i=1}^{n} \ln(P(\Theta_i[k] | s[k] = j)) + \ln(p_{ij}) + \alpha_{k-1}(i) \right]. \tag{7} \]
From this form, we can develop the diffusive algorithm; before doing so, we find it convenient to define some further notation. In particular, we use the notation \( \alpha_{k-1} \triangleq \begin{bmatrix} \alpha_{k-1}(1) \\ \vdots \\ \alpha_{k-1}(m) \end{bmatrix} \) and also use
\[ P(\Theta_i[k] | s[k]) \triangleq \begin{bmatrix} P(\Theta_i[k] | s[k] = 1) \\ \vdots \\ P(\Theta_i[k] | s[k] = m) \end{bmatrix}. \]

Finally, we are ready to specify a diffusive protocol that allows each sensor to compute \( \alpha_k \) from \( \alpha_{k-1} \) and the network parameters in a completely distributed manner. The protocol is specified in
three steps:

1) Each sensor \( l \) computes an initial estimate matrix \( Z_l[0] \). Sensors \( l = 2, \ldots, n \) compute their initial estimate matrices as \( Z_l[0] = n e_l \ln(P(\Theta_l[k] | s[k]))' \), where \( e_l \) is an indicator vector with entry \( l \) equal to 1. Also, Sensor 1 computes its initial estimate matrix as \( Z_1[0] = n e_1 \ln(P(\Theta_1[k] | s[k]))' + n \ln(P) + n \ln(\alpha_{k-1} I') \), where the natural logarithms are assumed to be taken elementwise. Notice that the initial estimate matrix for each sensor depends only on observations made by that sensor, as well as the global parameters \( P \) and (for Sensor 1) the results from the previous stage of the Viterbi algorithm.

2) An agreement protocol with agreement law \( \frac{1}{n} \ldots \frac{1}{n} \) and initial opinions \( Z_1[0], \ldots, Z_n[0] \) is applied. (To be precise, each entry of the matrices \( Z_1[0], \ldots, Z_n[0] \) is used as the initial opinion of an agreement protocol with agreement law \( \frac{1}{n} \ldots \frac{1}{n} \). Since the network graph is ergodic, we know that such a protocol can be developed.) In this way, each sensor asymptotically computes \( \overline{Z} \triangleq \frac{1}{n} \sum_{l=1}^{n} Z_l[0] \). Notice that the \( j \)th column of \( \overline{Z} \) contains the entries \( \sum_{l=1}^{n} \ln(P(\Theta_l[k] | s[k] = j')) + \ln(p_{ij}) + \alpha_{k-1}(i) \), for \( 1 \leq i \leq m \).

3) Each sensor finds \( \alpha_k(j) \) as the maximum entry of column \( j \) of the asymptotic matrix \( \overline{Z} \), and in turn determines the optimal state sequence estimate. Agent 1, in particular, thus has available \( \alpha_k \) for the next stage of the Viterbi algorithm.

Some further discussion of state sequence estimation algorithm for the DiHMM is useful:

- We note that each stage of the Viterbi algorithm for the DiHMM is analogous to the distributed MAP detection algorithm developed in Section 3 (and in fact can be viewed as a MAP detection problem). Hence, we note that the secondary task of providing each agent with the necessary network parameters can be achieved in an analogous manner to that used for MAP detection. One nominal difference is each agent seemingly requires the parameter \( n \) (the number of agents in the network) to compute its initial estimate matrix at each stage. However, we note that the optimal state sequence can be found even if the computation of multiplying by \( n \) to find the initial estimate matrices. In this case, the vectors \( \frac{\alpha_k}{n^k} \) would be
computed at each stage instead of $\alpha_k$; however, since the ordering of the entries in the vectors would remain the same, so would the state sequence estimate. This freedom to scale the computed probabilities is well-known for the centralized HMM estimation problem, but is of special importance for DiHMM estimation because the parameter $n$ may be difficult to obtain in a decentralized manner.

- It was noted in [6] that Kalman filtering can also be achieved using a diffusive protocol at each stage. We note that this distributed Kalman filtering algorithm is analogous to our DiHMM filtering problem, but with a continuous-valued rather than discrete-valued state sequence. We believe that our algorithm for HMM estimation, together with the analogous algorithm for Kalman filtering, constitute a first step toward an efficient and totally-distributed (leaderless) algorithm for filtering.

References


