HMM-Based Characterization of Channel Behavior for Networked Control Systems

Jian Chang
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA
jianchan@cis.upenn.edu

Krishna K. Venkatasubramanian
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA
vkris@cis.upenn.edu

Chinwendu Enyioha
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA
cenyioha@ee.upenn.edu

Shreyas Sundaram
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
ssundara@uwaterloo.ca

George J. Pappas
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA
pappasg@ee.upenn.edu

Insup Lee
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA
lee@cis.upenn.edu

ABSTRACT

We study the problem of characterizing the behavior of lossy and data corrupting communication channels in a networked control setting, where the channel's behavior exhibits temporal correlation. We propose a behavior characterization mechanism based on a *hidden Markov model* (HMM). The use of a HMM in this regard presents multiple challenges including dealing with incomplete observation sequences (due to data losses and corruptions) and the lack of *a priori* information about the model complexity (number of states in the model). We address the first challenges by using the plant state information and history of received/applied control inputs to fill in the gaps in the observation sequences, and by enhancing the HMM learning algorithm to deal with missing observations . Further, we adopt two model quality criteria for determining behavior model complexity. The contributions of this paper include: (1) an enhanced learning algorithm for refining the HMM model parameters to handle missing observations, and (2) simultaneous use of two well-defined model quality criteria to determine the model complexity. Simulation results demonstrate over 90% accuracy in predicting the output of a channel at a given time step, when compared to a traditional HMM based model that requires complete knowledge of the model complexity and observation sequence.

Categories and Subject Descriptors

B.4.5 [**Reliability, Testing, and Fault-Tolerance**]: Redundant Design; G.3 [**Probability AND Statistics**]: Markov Process; K.6.m [**Miscellaneous**]: Security

General Terms

Algorithms, Design, Reliability, Security

Keywords

Networked Control System, Majority Voting, Hidden Markov Model

1. INTRODUCTION

Networked control systems (NCS) are spatially distributed control systems where the communication between the plant sensors, plant actuators, and controllers takes place through a network [9]. However, the presence of a network within the control loop can adversely affect the performance of the system due to the inherent unreliability (*e.g.*, packet drops or transmission latency) of the communication channel. Such anomalies have direct consequences on the stability of the plant, as shown in previous studies [1, 2, 7, 8, 11, 17, 19].

As NCS have become more pervasive in large-scale industrial networks [22], the potential for data corruption due to unmitigated faults or malicious attacks increases considerably. However, an investigation of the combined effects of packet drops and data corruptions in the network in NCS has received limited attention. Our previous paper [24] studied a NCS with lossy and faulty (*i.e.,* data-corrupting) communication channels, and examined the use of a triple-modular-redundant channels. We use a majority voting scheme augmented with a simple channel behavior model to determine which channel inputs to apply to a plant within a given time-step. The goal is to tolerate a single data-corrupting channel and achieve mean square stability, assuming that the behavior of each communication channel can be modeled as an independent and identically distributed (i.i.d.) random variable.

This assumption of channel behavior being modeled as an i.i.d. random variable may not often hold in practice, especially in the case of wireless channels where burst errors or faults often persist over multiple time-steps [16]. Consequently, in this paper we focus on studying the scenarios where the behavior of each channel at the current time-step is *correlated* with its behavior in the previous t time-steps (for some unknown t). To this end, we adopt a hidden Markov model (HMM) framework to design a channel behavior characterization mechanism. The idea is to construct an HMM-based behavior model for each communication channel in the NCS. Given a channel's observed behavior sequence over time, the parameters of the corresponding HMM are computed and progressively refined. This HMM-based approach provides a probabilistic characterization of the channel's tendency to correctly or incorrectly transmit data at a given time-step. Using this behavior model, for an input received over a specific communication channel, one can make well-informed decisions about whether it should be applied to stabilize the plant. This is particularly useful in the event majority voting fails in a time-step due to the lossy and data corrupting nature of the channels (resulting in insufficient information to make a decision). Previous studies have focused on using HMMs as the reasoning engine to identify root causes of faults or erroneous states of the system [6, 10, 14, 21, 26–28]. In contrast, we adopt a black-box view that only focuses on identifying the correlation between a channel's current behavior and its behavior history. This allows us to propose a more general solution to the problem of dealing with unreliable channels in an NCS setting.

Designing an effective channel behavior characterization mechanism using HMMs (in the context of NCS) presents two important challenges: (1) *Incomplete and Uncertain Observations:* Due to the presence of data loss and corruption, existing majority voting schemes cannot always accurately discern the behavior of the communication channels at every time step. This introduces considerable incompleteness and uncertainty in the observation sequence and the channel's behavior model. (2) *Unknown Model Complexity:* It is critical to identify the number of states of the HMM behavior model (henceforth referred to as *model complexity*), which accurately captures a channel's behavior pattern. However, such information is often unknown *a priori*.

In addressing these challenges, we make two main *contributions*: (1) We design an enhanced learning algorithm for refining the HMM model parameters, which can handle missing observations. Additionally, we simultaneously reduce the missing observations by using the history of received and applied control inputs and the knowledge of the current plant state to fill the gaps in the observation sequences, with the benefit of hindsight. (2) We adopt two well-defined model quality criteria to determine the HMM complexity.

The rest of the paper is organized as follows. In Section 2 we discuss the system model and problem statement. In Section 3 we present the HMM-based channel behavior characterization mechanism followed by Section 4 where we present the enhanced HMM learning algorithm that can deal with missing data. In Section 5 we present our approach to determine the model complexity. In Section 6 we present the evaluation results and in Section 7, we conclude the paper.

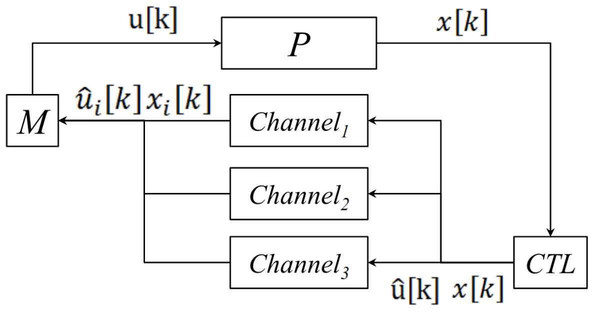


Figure 1: System Model

2. SYSTEM MODEL AND PROBLEM STATEMENT

Consider the networked control system shown in Figure 1. The plant P is given by the dynamical system

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \ , \tag{1}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are real-valued matrices of appropriate dimensions. $\mathbf{x} \in \mathbb{R}^n$ is the system state vector, $\mathbf{u} \in \mathbb{R}^m$ is the system input vector, and k denotes the time-step of the system.

To obtain the desired behavior from the plant, the state $\mathbf{x}[k]$ is sent to a controller CTL. Based on $\mathbf{x}[k]$ (or perhaps the entire history of past states), the controller computes an input $\hat{\mathbf{u}}[k]$ to apply to the plant. This value is then sent through an imperfect communication channel. In this paper, we consider three types of behaviors that a channel can exhibit at any given time-step: (1) *correct behavior (denoted as* `C`*):* the data is faithfully transmitted; (2) *faulty (data-corrupting) behavior (denoted as* `I`*):* the data being transmitted is modified due to unmitigated faults or malicious attacks; (3) *lossy (data-dropping) behavior (denoted as* `D`*):* the data is lost during the transmission.

To compensate for the faulty and lossy nature of the channels, we consider a triple-modular redundant scheme (see Figure 1). This scheme consists of three disjoint communication channels between the controller and the plant. The channels are used to send the control input $\hat{\mathbf{u}}[k]$ along with system state measurement $\mathbf{x}[k]$ to keep track of the plant dynamics based on the inputs applied over time. A *manager* component M receives values $\hat{\mathbf{u}}_i[k]$ ($i \in \{1, 2, 3\}$) sent through the three different channels and makes a decision on what input $\hat{\mathbf{u}}_i[k]$ to apply. To apply correct control inputs to the plant and to avoid corrupted ones, the manager M needs to properly discern a channel's behavior. By default, the manager implements a *majority voting* scheme for this purpose. With triple-modular redundancy, majority voting can be effective in the identification of the faulty channel when only one channel exhibits such behavior in a given time-step, and there are no packet losses. On the other hand, majority voting with lossy and faulty channels works as follows:

- The reception of at least two matching inputs at the manager in a given time-step results in its acceptance as a correct input (`C`) and its consequent application to the plant. Any channel whose value does not match this correct input is deemed as producing an incorrect input (`I`). Similarly, channels that drop the control input in a time-step are classified as dropped (`D`).
- If in a given time-step, two inputs are received and they do not match, or only one input is received then majority voting fails to find an appropriate input. In this case we use `X` to denote the behavior of such channels in that time-step (Example 1).

Example 1 *The true behaviors of the three input channels and the behavior types discerned using the majority voting scheme (`X` denotes the unknown behavior) is demonstrated.*

- *True behavior sequences:*

Channel 1	C C C C C C I D
Channel 2	C I C I I D I D
Channel 3	C C D D I D I D

- *Discerned behavior sequences using majority voting:*

Channel 1	C C C X X X X D
Channel 2	C I C X X D X D
Channel 3	C C D D X D X D

Due to the ambiguity introduced by packet drops and corruptions, we showed in [24] that a simple majority voting scheme may actually not provide stability if the channels drop packets with a sufficiently high probability (even when a single fault-free channel with the same probability of packet drop would be able to provide stability). We went on to demonstrate the use of a simple Bayesian estimator to improve the performance of majority voting by assuming that the behavior (both correct and incorrect) of each channel can be modeled as an i.i.d. random variable. We relax this assumption in this paper, since it is restrictive and may not hold in many practical situations.

Example 2 *For the Bernoulli packet-drop behavior model studied in previous works [8, 9, 20, 24], the probability distribution of possible behavior types is given below, where p is the data drop probability.*

$$\begin{cases} D & \text{with probability } p \\ C & \text{with probability } 1-p \\ I & \text{with probability } 0 \end{cases}$$

This model represents a case where the behavior at the current time-step is independent from the behavior type of the previous time-steps. In other words, its current behavior is correlated with behavior over the previous *zero* time-steps.

Example 3 *Consider a scenario where a channel alternatively exhibits the three types of behaviors described above. The observed behavior sequence Ω_S of this channel is:*

$$\Omega_S = C\ I\ D\ C\ I\ D\ C\ I\ D\ C\ I\ \ldots$$

In the above example, it is clear that the behavior type at the current time-step is correlated with (and actually completely specified by) the behavior in the previous *one* time-step. This behavior model is more general than the i.i.d. assumption made by previous works studying NCS [8, 9, 20, 24].

Problem 1 *In this paper, we study the problem of developing a robust and effective behavior characterization mechanism in the context of NCS, which focuses on the scenarios where a channel's behavior at the current time-step is* correlated *with its behavior in the previous* t *time-steps, for some unknown integer t.*

Remark: Note that we do not focus on the issue of *stability* in this paper. We assume that the system is designed in such a way that stability (in an appropriate sense) is attained despite the presence of packet drops (*e.g.,* if the conditions provided in works such as [7, 9, 15, 24], *etc.* are satisfied). Instead, our main goal is to design a *monitoring* mechanism to characterize the behavior and quality of the various channels, which can then be used by the plant operator to take appropriate actions, such as shutting down a severely malfunctioning channel. We will show how the hidden Markov model and its standard learning algorithms can be adapted and modified for effective channel behavior modeling, and to handle the particular complexities in the networked control context.

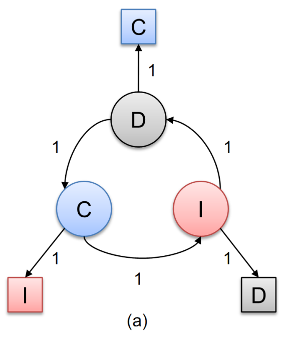
3. CHANNEL BEHAVIOR CHARACTERIZATION

In this section, we present the design of an effective behavior characterization mechanism for NCS using the hidden Markov model (HMM). Before delving into the details, we make the following important assumptions regarding the channels in our system model :

1. We consider a setup with *three* channels and assume each channel operates independently of the others; that is, there is no correlation between their behaviors at any time-step. Each of the channels can exhibit correct, faulty and lossy behavior.
2. We assume `I` inputs received from two or more channels at the same time-step will not agree under the majority voting scheme; only `C` (correct) inputs will match with each other.
3. There exists an appropriate Lyapunov function for the system that decreases whenever correct (`C`) inputs are applied; we assume that incorrect (`I`) inputs will not decrease this function.

3.1 Hidden Markov Model

To formally represent the class of behavior patterns discussed in Section 2, we adopt a hidden Markov model (HMM) framework [18]. A HMM consists of N states, denoted by



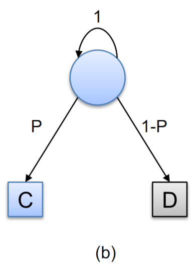


Figure 2: Examples of HMM Fault Model

the set $\Omega_X = \{q_1, \ldots, q_N\}$, and M outputs, denoted by the set $\Omega_O = \{v_1, \ldots, v_M\}$. The relationships between a fixed set of states and set of outputs is specified by two stochastic processes $\Lambda = \{E, F\}$. The first stochastic process describes the transition between states, whose parameters are encoded in the transition probability matrix $E = \{e_{ij}\}$, where $e_{ij} = P(\Omega_X(t+1) = q_j | \Omega_X(t) = q_i)$. The second stochastic process describes the relationship between each state and all the possible outputs, whose parameters are encoded in the emission probability matrix $F = \{f_i\}$, where $f_i = \{P(\Omega_O = v_1 | \Omega_X = q_i), \ldots, P(\Omega_O = v_M | \Omega_X = q_i)\}$. In a HMM, only the outputs (which are dependent on the states) are visible; the states and transitions between them are not directly observable, and are therefore *hidden*.

The HMM is an ideal modeling choice in our setting, since the temporal correlation between a channel's current behavior and its behavior history can be considered to be Markovian. As a result, we model the three possible behavior types of the channels as the output of the HMM. Furthermore, the behavior sequences with length t can be modeled using N states in the HMM, where $N = 3^t$ (this corresponds to the combination of all possible behavior sequences with length t). For instance, as illustrated in Figure 2(a),[1] the HMM of Example 3 has $3^1 = 3$ states, to model the correlation between its current behavior and its behavior in the previous $t = 1$ time-step. Similarly, for Example 2, the corresponding HMM has $3^0 = 1$ state to model the correlation between its current behavior and behavior of the previous $t = 0$ time-steps, as shown in Figure 2(b).

HMMs have traditionally been used to answer two main questions: (1) given the observed sequence, estimate the most likely model parameters of the hidden model Λ; and (2) given the model parameters Λ and the observed sequence, find the most likely state sequence up to the current time-step and the most likely output at a given time-step. The Baum-Welch algorithm [25] is used as a means to achieve the former, efficiently estimating the model parameters in two passes, iteratively. The first pass goes forward in time and computes a conditional probability $P(q_i|\Omega^S_{1:k})$ of ending up in any particular state q_i given the first k outputs in the observation sequence Ω_S; while the second goes backward in time and computes the conditional probability $P(\Omega^S_{k:t}|q_i)$ of seeing the remaining observations in the sequence given any state q_i as the starting point. These two types of conditional probabilities can then be combined to obtain the distribution over states at any specific point in time given the entire observation sequence. Similarly, the latter question, studied in [12], is addressed by computing the conditional probability $P(q_i|\Omega^S_{1:k})$ to find the most likely state sequence. By combining this conditional probability with the emission probability matrix, one can further obtain the most likely output at a given time-step.

[1]The state transitions and output emissions with 0 probability are omitted.

Algorithm 1 Behavior Characterization Scheme

```
1: Set value of the variable $Model_C$ for each channel, based on a priori over-estimation of the number of states in the HMM
2: **if** The majority voting scheme finds matching inputs **then**
3:     Correctly identify the behavior type for each channel
4:     Append the observed behavior to the behavior history of channels
5:     Refine missing observations in the behavior history with hindsight
6:     **if** Sufficient observations are accumulated for a channel **then**
7:         Execute the model complexity estimation algorithm to update the value of variable $Model_C$ for that channel
8:     **end if**
9:     Update each channel's behavior model based on the $Model_C$ value and apply traditional HMM learning algorithm
10: **else**
11:     Use each channel's behavior model to predict its behavior type at this time-step
12:     Update each channel's behavior model using an enhanced HMM learning algorithm
13: **end if**
```

3.2 Overview of Behavior Characterization

In this paper, we use hidden Markov Model as the basis for characterizing the channel's behavior, given the observation sequence. Applying this established technique in our setting is, however, non-trivial. There are two major challenges: (1) the behavior history may contain missing entries (which are denoted as X) due to the limitation of the majority voting scheme. Since this unknown behavior type X is not contained in the output set $\Omega_O = \{\mathtt{C}, \mathtt{I}, \mathtt{D}\}$ of the HMM, the classic learning algorithm for HMMs cannot properly learn the channel behavior; and (2) the number of the HMM states is also not known *a priori* in our setting.

A skeletal picture of our channel behavior characterization mechanism is shown in Algorithm 1. The behavior characterization is implemented as a part of the manager block M in the NCS and operates in six steps:

- **Line 1**: Start with an estimate of the complexity of each channel's behavior model (*i.e.,* the number of states in the HMM required to describe the channel's behavior pattern) as its input. Such an estimate can be obtained based on application-specific knowledge, or from experience[2]. During the early phase of our be-

[2]It is important to note that an accurate estimation of such model complexity may not be easy to obtain in some applications, so one would prefer to have an over-estimated the value, trading-off computational expense for model expressiveness.

havior characterization mechanism (when only limited observations of channel behavior are available), this model complexity estimate will be used as the basis for constructing the behavior model for each channel.

- **Lines 2-4**: Observe the data received from the channels at the current time-step k and discern the corresponding behavior types (*i.e.,* `C`, `I`, or `D`) using majority voting.
- **Line 5**: If majority voting succeeds at time-step k, then for all time-steps j, $0 < j \leq k-1$, where majority voting was not successful in assessing the channels' behavior (*i.e.,* due to incomplete information), use the current state information (which has been correctly inferred via majority voting), and the history of received and applied control inputs to infer the behavior of the channels up to time-step j. (see Section 4)
- **Lines 6-8**: Once sufficient observations of channel behaviors are accumulated, use the model complexity estimation algorithm to update the estimate of the model complexity ($Model_C$) (see Section 5).
- **Line 9**: Use the observed channels' behavior sequence to refine the correspond HMM-based behavior model using the estimated $Model_C$ value as its parameter.
- **Lines 11-12**: If the majority voting scheme fails to make a decision at time-step k, we use the behavior model of each channel to predict its behavior in that time-step. An enhanced HMM learning algorithm is used, which considers the true behavior types of the corresponding channels as unknown at this time step, and updates the corresponding behavior models appropriately, as we shall see in the Section 4. This continues until a future time-step $k+l$, when the majority voting scheme succeeds (restart at Line 2)

4. HANDLING INCOMPLETE OBSERVATION SEQUENCE

We propose two countermeasures to address the limitations of majority voting in providing a complete observation sequence: (1) We design a scheme to reduce the number of time-steps in which the behavior is unknown. (2) We adopt a modified version of the Baum-Welch algorithm as our enhanced HMM learning algorithm, similar to the approach proposed in [29], for handling the missing data in the observation sequence.

4.1 Reducing Incompleteness of Observation Sequence

The reducing of incomplete observation sequence is a five step process. They include: (1) the controller C sends both the control input $\hat{\mathbf{u}}[k]$ and the system state measurement $\mathbf{x}[k]$ over the network ; (2) the manager keeps the history of all the control inputs received from the channels and the ones applied to the plant; (3) if the majority voting scheme has matching data at time-step k, we obtain the correct system state $\mathbf{x}[k]$; (4) the dynamical model shown in Equation (1) is then used to recover the state of the plant at the previous points in time when the state was unknown due to lack of information for majority voting; and (5) for every `X` before time k, the manager uses the recovered state to determine whether the inputs provided by the channels at that time step would have decreased the Lyapunov function — if so, the input will be labeled as `C` (under the assumption that `I` inputs won't drive the plant state towards stability).

Step (5) in the above sequence can be accomplished if the plant dynamics satisfy certain conditions. For instance, since

$$\mathbf{x}[k] = \mathbf{A}^T\mathbf{x}[k-T] + \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{T-1}\mathbf{B}\end{bmatrix}\begin{bmatrix}\mathbf{u}[k-1]\\ \mathbf{u}[k-2]\\ \vdots \\ \mathbf{u}[k-T]\end{bmatrix}$$

for any nonnegative integer T, one can recover $\mathbf{x}[k-T]$ from the above expression if $\mathbf{A}^T$ is invertible (or equivalently, if $\mathbf{A}$ is invertible).

Remark: The aforementioned improved mechanism can also be applied to a system with nonlinear dynamics

$$\mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{u}[k]) \tag{2}$$

as well, where the goal is to determine the state $\mathbf{x}[k-T]$ given $\mathbf{x}[k]$. Inspired by [13, 23], the extraction of a previous state given the current state can be done by simultaneously solving a set of nonlinear equations, provided that the nonlinear dynamics (2) satisfy certain properties.

4.2 Learning with Incomplete Observation Sequence

In the absence of a successful majority voting in a time-step, it is possible that observation sequence from a channel has missing elements. In [29], the authors propose an enhanced model parameter learning algorithm for hidden semi-Markov models with missing data in the observation sequence, and use it for mobility tracking applications. In this paper, we adopt a similar approach and design an algorithm that handles missing observations for HMM by modifying the Baum-Welch algorithm. In the original Baum-Welch algorithm, the iterative computation of the forward and backward probabilities require the use of the emission probability matrix. However, the emission probability for unknown behavior type `X` is not defined. To mitigate this issue, the idea is to consider all possible behaviors that can be exhibited while updating the model parameters when the unknown feedback `X` is encountered. In this paper, we consider `X` to be either `C` or `I` with equivalent probability values.[3] Therefore, the emission probability for `X` can be interpreted as the combination of the emission probability of `C` and `I`. The rest of the Baum-Welch algorithm remains unchanged.

5. MODEL COMPLEXITY ESTIMATION

Initially, we use an over-estimation of the number of states in the HMM for modeling a channel's behavior pattern. We then use a model complexity estimation algorithm to accurately determine the actual number of states with limited observation of channel behavior. In the algorithm, we adopt two model quality criterion: (1) *Condition number* γ, which is defined as $\gamma = \sigma_{min}(E|F)/\sigma_{max}(E|F)$, where σ_{min} and σ_{max} are the smallest and largest singular values of the HMM parameter concatenation matrix $E|F$, respectively [3]; and (2) *Observation probability*, which is defined

[3]Since feedback type `D` can always be confidently discerned, we do not need to consider it as a possibility.

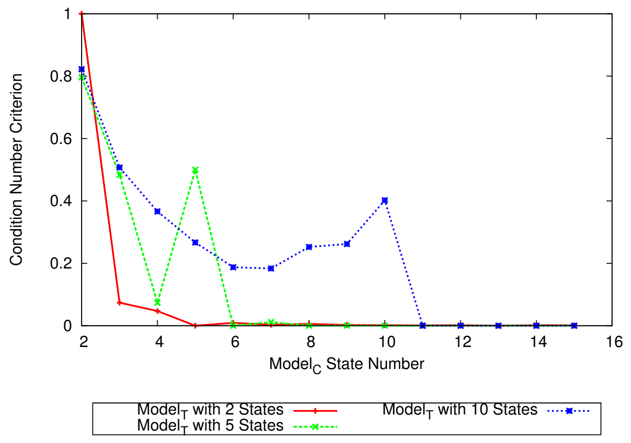


Figure 3: Condition Number Variability

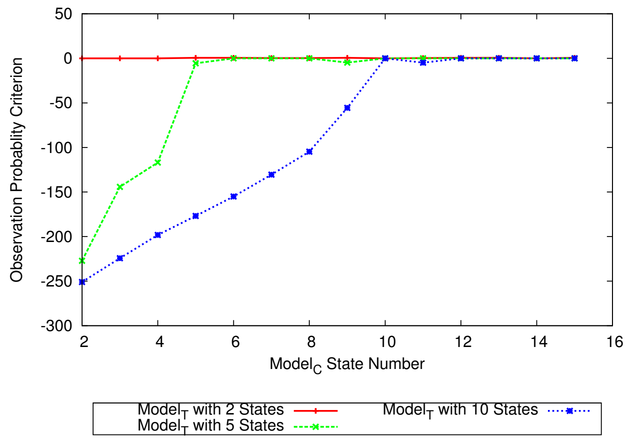


Figure 4: Observation Probability Variability

as the probability for the learned HMM model to generate the observed behavior sequence.

5.1 Condition Number

The condition number criterion determines whether certain states in the model are redundant or unnecessary. Figure 3 shows the variation in the condition number with respect to candidate model complexity (referred to as $Model_C$). The figure shows the results for three scenarios where the ground truth for actual number of states in the model (referred to as $Model_T$) are 2, 5 and 10 respectively. Irrespective of the $Model_T$ value, the pattern observed hold consistently:

- When the $Model_C \leq Model_T$, the condition number of the candidate model first decreases and then increases, with a local maxima at $Model_C = Model_T$. However, the condition number values are still relatively high, even at local minimas. The reason here is that when the model is not expressive enough, it can only capture certain sub-patterns within the entire underlying behavior pattern, until $Model_C = Model_T$.
- When $Model_C > Model_T$, the condition number of the candidate model drastically decreases and tends toward 0. This reflects the fact that certain states in the candidate model are redundant (*i.e.,* unnecessary for capturing the ground truth behavior pattern).

5.2 Observation Probability

The observation probability criterion indicates how well the learned HMM model captures the observed behavior sequence. We illustrate this criterion in Figure 4. Similar to the condition number criterion, Figure 4 shows variation in the observation probability with respect to $Model_C$. The figure shows the results for three scenarios where $Model_T$ is set to 2, 5, and 10, respectively. Irrespective of the $Model_T$ value, again the patterns we observed hold consistently:

- When the $Model_C \leq Model_T$, the observation probability of the candidate model monotonically increases, and reach its maximum value when $Model_C = Model_T$.
- When $Model_C > Model_T$, the observation probability value remains mostly unchanged. The reason here is that as the value $Model_C$ increases, the behavior pattern captured by the candidate behavior model is increasingly closer to the ground truth, until $Model_C = Model_T$ when the observation probability criterion reach its maximum value. And when $Model_C > Model_T$, the candidate model complexity is an over-kill for modeling the ground truth behavior pattern, and the observation probability value cannot increase further.

In other words, using a candidate model with $Model_C > Model_T$ (*i.e.,* an over-estimation of the behavior model complexity), the expressiveness of candidate model is good enough to capture the underlying behavior pattern, the only drawback is that it requires additional computational resources and potentially more observation data to train the candidate model, compared to the ideal ground truth.

5.3 Combined Criterion

By combining these two criterion, we can design a model complexity estimation algorithm that achieves two goals: (1) output an accurate estimation for $Model_T$; (2) achieve the first goal with "limited" tries of different candidate models. A pseudocode description of our model complexity estimation algorithm is shown in Algorithm 2. The overall algorithm consists of two stages. In the first stage, we attempt to identify a proper lower and upper bound of candidate model complexity by utilizing the condition number criterion (line 2 - 13). The idea is to exponentially increase the lower and upper bound for every iteration (line 4 and line 11), until we successful detect the drastic change of the condition number value between the two bounds (line 7). As we discussed previously, such a pattern indicates that the true model complexity lies between the lower bound and upper bound value. We record these bounds, as well as the observation probability value of the upper bound model to be used in the later stage (line 8-9). In the second stage, we attempt to find the smallest value for $Model_C$ that can achieve the highest observation probability value (*i.e.,* as the upper bound model) using a binary search scheme [4] (line 16-30). Let the range between the two bounds be n, then the best, worst and average time complexity of the algorithm is $O(1)$, $O(logn)$, and $O(logn)$, respectively.

Algorithm 2 Model Complexity Estimation Algorithm

```
1:  Initialize variable $i$ to be 1
2:  while TRUE do
3:    Set the lower bound ($LB$) and upper bound ($UB$) of candidate model state number to be $2^i$ and $2^{(i+1)}$, respectively
4:    Learn two HMM models with state number to be the lower bound and upper bound, respectively
5:    Compute the condition number and observation probability criterion for the two models, denote these values as $CN_{LB}$, $OP_{LB}$, and $CN_{UB}$, $OP_{UB}$, respectively
6:    if $CN_{LB} >> CN_{UB}$ then
7:      Record the observation probability value $OP_{UB}$
8:      Output $LB$, $UB$
9:      Break
10:   else
11:     $i = i + 1$
12:     Continue
13:   end if
14: end while
15:
16: Set variable $s = \lfloor((LB + UB)/2)\rfloor$, that is the largest integer value smaller than or equal to $(LB + UB)/2$
17: while TRUE do
18:   Learn a candidate HMM model with state number $s$, and compute observation probability of it, denote as $OP_S$
19:   if $OP_S \simeq OP_{UB}$ then
20:     Set $LB = s$;
21:     Set $s = \lceil((LB + UB)/2)\rceil$, that is the smallest integer value larger than or equal to $(LB + UB)/2$
22:   else
23:     Set $UB = s$;
24:     Set $s = \lfloor((LB + UB)/2)\rfloor$;
25:   end if
26:   if $LB \equiv UB$ then
27:     Output $s$
28:     Break
29:   else
30:     Continue
31:   end if
32: end while
```

6. EVALUATION

It has been proved in the literature that the HMM learning algorithm will converge to the ground truth with infinite observations of the behavior sequence [5]. In practice, one needs to train the model with "sufficient" observation data to achieve good convergence. Similarly, one requires sufficient data for our model complexity estimation algorithm to work effectively. The sufficiency of observation data is highly dependent on the model complexity and often needs to be determined empirically. In Figure 5, we present our results from numerical experiments. We run the HMM learning algorithm 20 times for different $Model_T$ values (*i.e.,* as shown by the x-axis in Figure 5). For each run, we gradually increase the length of the observation sequence and record the length when the candidate model complexity converges to $Model_T$. The median, minimum and maximum value of the training sequence length for different model complexity values are illustrated in Figure 5. As a result, to obtain an accurate model complexity estimation (line 6-8 in Algorithm 1), the observation sequence length should be greater than some minimum threshold value of the training sequence length required for the chosen candidate model complexity. In practice, to obtain such threshold values for different model complexities requires conducting empirical experiments as we discuss above.

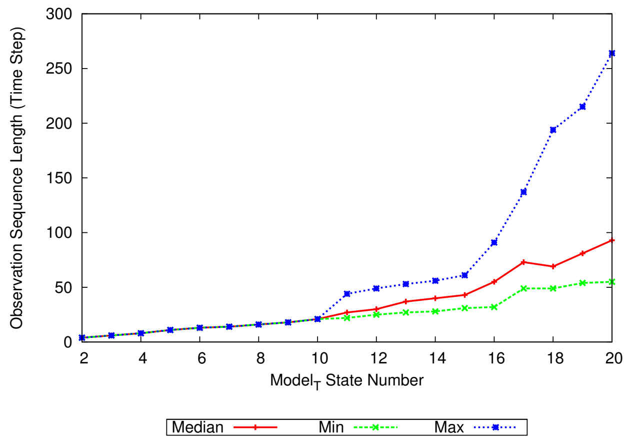


Figure 5: Amount of Observation Data Sufficient for various $Model_T$

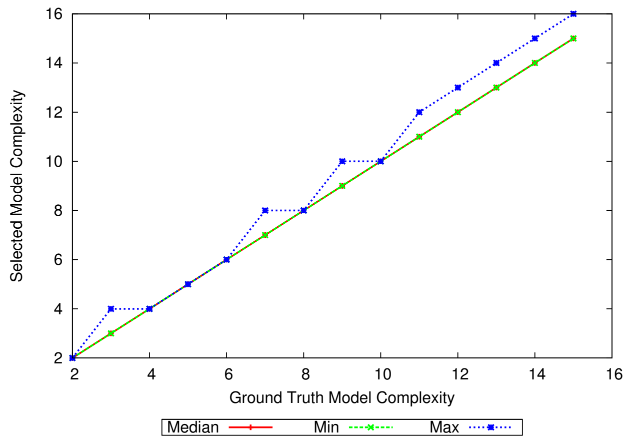


Figure 6: Effectiveness of Model Complexity Estimation

In Figure 6, we show the effectiveness of our model complexity estimation algorithm. We ran the algorithm 20 times for different $Model_T$ values (as shown as the x-axis in Figure 6) with sufficient data (*i.e.,* more than the minimum value for each complexity level, as shown in Figure 5). The Median, Min, and Max value of the model complexity estimation results are reported. We can see that the algorithm never under-estimates the model complexity, which is desirable for maintaining the accuracy of the overall behavior characterization scheme. Further, the algorithm only occasionally over-estimates the model complexity, and the difference between the over-estimated model and the ground truth model is acceptably small (often only one state more for the over-estimated model).

We then evaluated the effectiveness of our proposed scheme

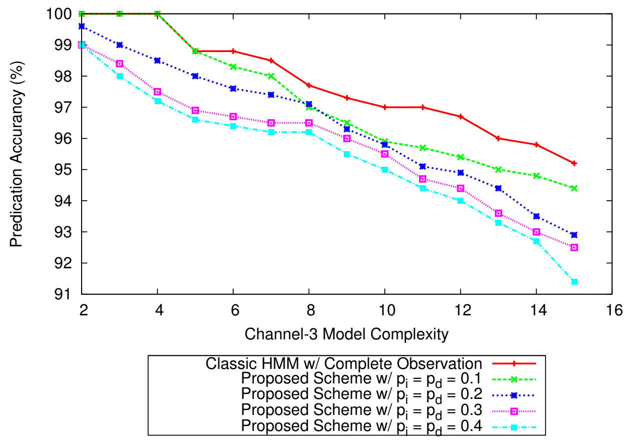


Figure 7: Proposed Approach Effectiveness (Median Value)

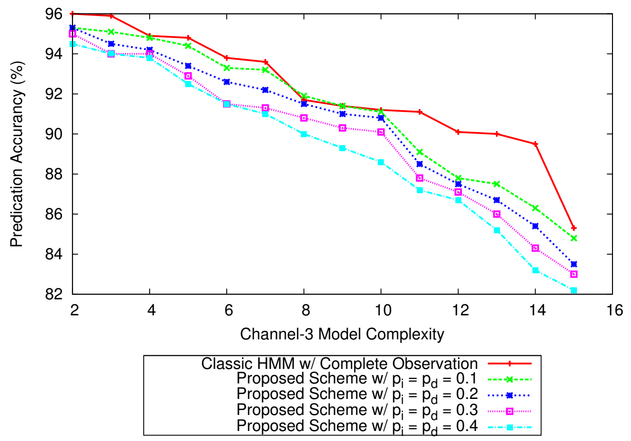


Figure 8: Proposed Approach Effectiveness (Minimum Value)

(as described in Algorithm 1) using simulation. In the experiment, we have two communication channels with their behavior modeled with a Bernoulli distribution with packet drop rate p_d and data-corruption rate p_i. Additionally, we have a third channel whose behavior model is temporally correlated. We set the steady-state probability of packet-drop and data-corruption rates for the third channel to be 33%, respectively. The goal is to vary the values p_d and p_i of the two Bernoulli channels (from 10% to 40% as shown by different curves in Figure 7 and Figure 8), and control the percentage of unknown observations in the behavior sequence when applying the majority voting scheme. Then, we vary the model complexity of the third channel (from 2 to 15 states as shown by the x-axis in Figure 7 and Figure 8) and apply our model complexity estimation algorithm, after accumulating enough observations (as discussed in Section 5). By running the simulation 20 times with 1000 time-steps each, we measured the predication accuracy of the learned behavior model for the third channel at each time-step the majority voting scheme failed. For comparison purposes, we run a classic HMM learning algorithm in parallel. The algorithm is aware of the true model complexity value and has access to complete channel behavior observations (without unknown observations).

Figure 7 and 8 show the results of this experiment. Overall, our proposed scheme can achieve very high predication accuracy when compared to the classic HMM algorithm with complete observation sequence. We observe median prediction accuracy to be greater than 90% for most cases (see Figure 7), and minimum values greater than 80% (see Figure 8). The overall downward trend of the graph is because as the model complexity increases, the differences between the learned model and the true model increases as well.

7. CONCLUSION

We studied the problem of characterizing the behavior of communication channels in a networked control system, where the channel's behavior exhibits temporal correlation. We proposed a behavior characterization mechanism based on the hidden Markov model. We showed that there are many challenges in adopting this approach due to incomplete observations and the lack of *a priori* information about the model complexity. Further, we demonstrated that these challenges can be addressed by: 1) designing an enhanced learning algorithm for refining the HMM model parameters, which can handle missing observations. Additionally, we simultaneously reduce the missing observations by using the history of received and applied control inputs and the knowledge of the current plant state to fill the gaps in the observation sequences, with the benefit of hindsight; and (2) using two well-defined model quality criteria simultaneously to determine the HMM complexity. In the future, we intend to study and characterize stability of the system as the manager M in Fig. 1 switches between different (correct and incorrect) channels to apply a control input to the plant, as it learns their behavior model.

References

[1] Saurabh Amin, Alvaro Cardenas, and S. Sastry. Safe and secure networked control systems under denial-of-service attacks. In Rupak Majumdar and Paulo Tabuada, editors, *Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 31–45. 2009.

[2] A. Bemporad, W.P.M.H. Heemels, and M. Johansson (eds). *Networked Control Systems*, volume 406. Lecture Notes in Control and Information Sciences, Springer-Verlag, 2010.

[3] Terry Caelli and Brendan McCane. Component analysis of hidden markov models in computer vision. In *Proceedings of the 12th International Conference on Image Analysis and Processing*, pages 510–, Washington, DC, USA, 2003. IEEE Computer Society.

[4] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd edition, 2001.

[5] Brian G. and Leroux. Maximum-likelihood estimation for hidden markov models. *Stochastic Processes and their Applications*, 40(1):127 – 143, 1992.

[6] M Ge. Hidden markov model based fault diagnosis for stamping processes. *Mechanical Systems and Signal Processing*, 18(2):391–408, 2004.

[7] V. Gupta, A. F. Dana, J. Hespanha, R. M. Murray, and B. Hassibi. Data transmission over networks for estimation and control. *IEEE Transactions on Automatic Control*, 54(8):1807–1819, Aug. 2009.

[8] C. N. Hadjicostis and R. Touri. Feedback control utilizing packet dropping network links. In *Proc. of the 41st IEEE Conference on Decision and Control*, pages 1205–1210, 2002.

[9] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proc. of the IEEE*, 95(1):138–162, Jan. 2007.

[10] J Huang and P Zhang. Fault diagnosis for diesel engines based on discrete hidden markov model. *2009 Second International Conference on Intelligent Computation Technology and Automation*, pages 513–516, 2009.

[11] O. C. Imer, S. Yuksel, and T. Basar. Optimal control of LTI systems over unreliable communication links. *Automatica*, 42(9):1429–1439, Sep. 2006.

[12] G. D. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268 – 278, March 1973.

[13] M. Krstic, I. Kanellakopoulos, and PV Kokotovic. Nonlinear design of adaptive controllers for linear systems. *Automatic Control, IEEE Transactions on*, 39(4):738–752, 1994.

[14] J Lee, S Kim, Y Hwang, and C Song. Diagnosis of mechanical fault signals using continuous hidden markov model. *Journal of Sound and Vibration*, 276(3-5):1065–1080, 2004.

[15] Michael D. Lemmon and Xiaobo Sharon Hu. Almost sure stability of networked control systems under exponentially bounded bursts of dropouts. In *HSCC*, pages 301–310, 2011.

[16] A.S. Matveev and A.V. Savkin. Comments on control over noisy channels and relevant negative results. *Automatic Control, IEEE Transactions on*, 50(12):2105 – 2110, Dec. 2005.

[17] A.R. Mesquita, J.P. Hespanha, and G. Nair. Redundant data transmission in control/estimation over wireless networks. In *Proc. of the 2009 American Control Conference*, pages 3378–3383, 2009.

[18] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4 –16, Jan 1986.

[19] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proc. of the IEEE*, 95(1):163–187, Jan. 2007.

[20] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *Proc. of the 2001 American Control Conference*, pages 1491–1496, 2001.

[21] P. J. Smyth. Hidden Markov models for fault detection in dynamic systems. *NASA STI/Recon Technical Report N*, 933:30413, April 1993.

[22] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ICS) security. Technical Report 800-82, National Institute of Standards and Technology, Sep. 2008.

[23] J. Stumper and R. Kennel. Inversion of linear and nonlinear observable systems with series-defined output trajectories. In *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pages 1993–1998. IEEE, 2010.

[24] Shreyas Sundaram, Jian Chang, Krishna K. Venkatasubramanian, Chinwendu Enyioha, Insup Lee, and George J. Pappas. Reputation-based networked control with data-corrupting channels. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, HSCC '11, pages 291–300, 2011.

[25] Lloyd R. Welch. Hidden Markov Models and the Baum-Welch Algorithm. *IEEE Information Theory Society Newsletter*, 53(4), December 2003.

[26] Wee Chin Wong and Jay H. Lee. Fault detection and diagnosis using hidden markov disturbance models. *Industrial & Engineering Chemistry Research*, 49(17):7901–7908, 2010.

[27] Ren-Wu Yan and Jin-Ding Cai. Application of hidden markov model to fault diagnosis of power electronic circuit. *2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis*, pages 1–4, 2009.

[28] Jie Yu. Multiway discrete hidden markov model-based approach for dynamic batch process monitoring and fault classification. *AIChE Journal*, 2011.

[29] Shun-Zheng Yu and Hisashi Kobayashi. A hidden semi-markov model with missing data and multiple observation sequences for mobility tracking. *Signal Process.*, 83:235–250, February 2003.