IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 December 2023, accepted 19 January 2024, date of publication 23 January 2024, date of current version 1 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357608

RESEARCH ARTICLE

Obtaining the Most Likely Path in Stochastic Hidden Input Automata by Using Limited Optimal Discrete Control

METE ÖZBALTAN[1] **AND MEHMET KURUCAN**[2]
[1]Department of Computer Engineering, Erzurum Technical University, 25050 Erzurum, Turkey
[2]Department of Computer Engineering, Ardahan University, 75002 Ardahan, Turkey
Corresponding author: Mete Özbaltan (mete.ozbaltan@erzurum.edu.tr)

ABSTRACT We propose a new modeling framework to compute the most likely path for stochastic hidden systems; where the computation is based on the control theory of discrete event systems. The main innovation in this proposed model is calculating which event will have a higher probability of occurring in the future by applying k-step to the likelihood of events occurring at discrete times, which will give us the best way to transition between situations. We encode the problem as a node built with synchronous data-flow equations; then we apply the synthesis algorithm to the node in order to generate a controller that will find the most likely state sequence; where the algorithm is limited to a sliding window of a fixed number of discrete steps. We experimentally evaluate and validate our approach by comparing it with several algorithms, which are the most common and suitable algorithms applied for the best path calculation.

INDEX TERMS Baum-Welch algorithm, discrete controller synthesis, hidden systems, optimal control, Viterbi algorithm.

I. INTRODUCTION

The most used models among stochastic hidden systems are Hidden Markov Models (HMM), which is a generalized form of the Markov chain by latent the Markov process. The state transition matrix is represented by parameter A of the HMM, and the state observation densities are represented by parameter B. These parameters are estimated using Baum-Welch algorithm [17], [26].

The main challenge is finding the most likely path for such stochastic models. The proper solution for this problem is to select the option with the highest likelihood. Viterbi algorithm, a dynamic programming strategy, is one of the most effective methods used for this process. Viterbi algorithm takes the highest probability value of the transitions in the Markov graph to determine the most likely path. Calculating the most probable path gives a much better result especially when estimated parameters are used. As a result, we first estimate the HMM's state transition matrix and emission matrix using the training approach, and then we utilize the most recent estimates of those parameters to perform the Viterbi calculation.

The study on heart rhythm classification with recurrent neural networks (RNNs) and attention mechanisms achieved a state-of-the-art F1 score of 0.79 on an unseen test set [30]. ViterbiNet, a symbol detector integrating deep neural networks into the Viterbi algorithm, demonstrated robustness to channel state information uncertainty, showcasing its potential for seamless integration into communication systems [32]. The study presenting a high-performance Viterbi algorithm with bitslicing achieved superior throughput on GPU platforms, with implementations outperforming prior works and achieving 21.41 and 8.24 Gbps for hard and soft-decision cases, respectively [22]. The proposal of a fastIMM-extended Viterbi algorithm for real-time tracking of maneuvering targets using a ballistic acoustic array, incorporating α-β and α-β-γ filters for efficiency, demonstrated practicality and high efficiency in MATLAB simulations compared to various IMM target tracking methods [13]. The paper introducing SIEVE

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

(Space Efficient Viterbi) addressed scalability limitations in speech recognition on limited-memory devices, significantly reducing memory usage without introducing a runtime overhead, as demonstrated in experimental evaluations [10].

Furthermore, the control methods based on artificial intelligence represent an emerging technology and have shown certain effectiveness. The paper proposes a system for efficient sensing data offloading using multiple coordinated UAVs controlled by a base station, optimizing trajectories and network formations for improved efficiency. It demonstrates enhanced energy efficiency and delay performance compared to baseline methods [16]. Another study introduces a reinforcement learning-based control algorithm for VTOL aircraft, treating tracking control as a Markov decision process. The incorporation of wind fields and a quantum-inspired experience replay strategy ensures accuracy and robustness in various experiments [21]. Addressing GPS-denied environments, a paper presents a solution for precise positioning in multi-UAV close-formation flight using Lidar-based localization, k-means center point calculation, and a reinforcement learning-based formation control algorithm, showing efficient and robust performance [6]. Lastly, an innovative incremental reinforcement-learning-based algorithm for UAV tracking control in dynamic environments is introduced, employing a Markov decision process, policy relief, and significance weighting for high accuracy, effectiveness, and robustness [20].

In a multidisciplinary landscape such as robotics, control systems, communication networks, and software engineering this study shows an important effort. This research advances the probability of the path estimation within hidden stochastic systems by melting the discrete controller synthesis (DCS) approach with Hidden Markov Models (HMM) into the same pot. Thus, it earns essential contributions to achieve complex real-world challenges. In particular, the proposed method arises as an activator for designing controllers experts in navigating complicated terrains with an enriched grasp of likely tracks within the area of robotics. Relevantly, the integration of DCS methodologies suggests the prospect of optimized path adjustment to increase overall efficiency. This study exhibits its significance in the area of software engineering by providing controllers that support program correctness and safety by further expanding its relevance. The central motivation for this research derives from the aim to enhance accuracy by merging DCS principles with HMM methodologies.

A different approach is discrete controller synthesis (DCS), which can be used on the same model for the same job. Viterbi is a regularly used technique for determining the most likely path in a hidden stochastic system. However, this algorithm has also drawbacks as usual methods where the systems have especially long-term dependencies. Our proposed method makes this drawback an advantage by taking into account the DCS framework to provide a solution to this problem. The power of DCS around this context, it provides a strategy that states the controller how to choose among several possible processes with formal correctness by taking into account fundamental interaction between the system and the environment [1]. This approach allows greater control over the behavior of the system. This way puts it into the advantage of being able to consider complex specifications beyond simply finding the most probable path. According to this aspect, DCS is used as a powerful controller in well-known research areas such as robotics, control systems, communication networks, and software engineering for designing controllers to attain particular performance and safety requirements [2]. That being so, this tool, DCS, can be used as a powerful controller where it can effectively manage complex systems.

In consideration of the power of the DCS the innovation of our proposed method is shaped over the ability to gather information from the future for addressing the same task. The proposed method uses multiple steps to anticipate the maximum likelihood of a future event even though the common methods use single steps (i.e. k=0) for calculating events occurring at discrete time intervals. This method provides significantly more accurate results compared to the methods commonly employed currently. However, it has a higher computational complexity.

Discrete control synthesis (DCS) is a formal framework that provides the desired system behavior in line with the desired control objectives on an initially uncontrolled system by restricting controllable input variables. DCS technique is used to manage complex systems with formal correctness in many fields such as robotics, control systems, communication networks, and software engineering. For example: [19] construct controllers via the DCS technique for robotic systems to provide precise control and stable motion; [11] build controllers for communication networks that provide dependable and effective data flow; [18] create controllers using the DCS technique, that guarantee the safety and correctness of programs for software engineers. DCS is an effective approach preferred to overcome many problems in real-world tissues that require formal correctness.

Even if it is known the traditional techniques are demonstrated ability, like the Viterbi algorithm, however, their impact factor decreases when they face a complex or long time being touched by real-world conditions. The weakness of the common techniques that are faced when the problem is more complex produces a gap related to finding a solution for the most probable path problem. This new advanced study not only improves solutions in the area of hidden probabilistic systems but also leads fundamentally to scientific expression. By synthesizing DCS and HMM areas, the study shows a formalized framework that generates controllers like mastered type at adhering to strict behavioral and specification criteria. Especially, the k-step optimization that is shown here improves reliability and offers a flexible solution that can be used for various complexity levels with better accuracy than the Viterbi method.

The rest of this paper is organized as follows. Section II gives a basic definition of HMM and the estimation algorithm Baum-Welch. In Section III we show the principles of DCS and present our model and the objectives for hidden systems that are equipped with the DCS. In Section IV we show the obtained results for both models, and in the last Section V we conclude the whole paper with future works.

II. DEFINITION OF HMM

The HMM is a powerful sequence analysis method where it is widely used for modeling discrete time series data. It is a useful tool in fields of speech recognition [9], stock market analysis [23], and bioinformatics [31].

The model emerged as a stochastic model that contains a sequence of observations (or visible variables) in which the distribution of the unobserved Markov process generates. An HMM has two kinds of parameters. The first parameter is the state transition matrix (*i.e.,* $A(S_t, S_{t+1})$). It specifies the probability of transition from the current state to the next state. This process satisfies the first order of the Markov property where the next hidden state at time $t+1$ depends only on the current hidden state at time t [7]. The second parameter of the model is the emission matrix (*i.e.,* $B(S_t, o_t)$). It keeps the probability of emitting output symbols from a particular hidden state. This process also meets the first order of the Markov property where the corresponding output symbol of the particular hidden state is independent of all other hidden states and output symbols [15].

FIGURE 1 shows the dependencies of a HMM. The hidden states which are represented as circles and the output symbols are shown as squares. The figure shows the particular hidden state emits an output symbol at each discrete time t. Some studies present the initial distribution vector (*i.e.,* π) where the probabilities of initial distribution are held in. However, we have only one initial state and its initial probability is 1. Depending on this knowledge, we do not consider the initial vector in our work.

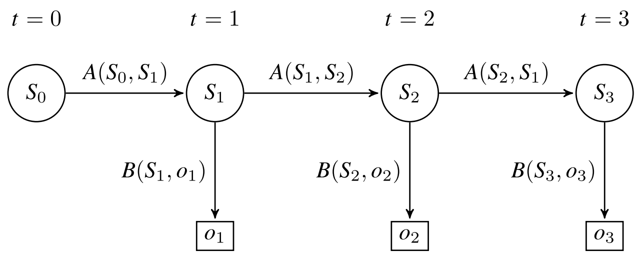


FIGURE 1. The graphical representation of an HMM structure.

The element of an HMM can be defined as a tuple $\theta = (S, O, A, B)$ where

- S is the set of hidden states in the model
- O is the set of observation symbols in the model
- A is state transition probability distribution where $A(q, q')$ stores the probabilities of state q' following state q:

$$A(q,q') = Pr(S_{t+1} = q'|S_t = q), q, q' \in S$$
$$\sum_{q' \in S} A(q,q') = 1 \quad \forall q \in S \tag{1}$$

- B is emission symbol probability distribution where $B(q, o)$ stores the probability of observation symbol o being produced by the hidden state q, discrete time t:

$$B(q,o) = Pr(o_t = o|S_t = q), o \in O, q \in S$$
$$\sum_{o \in O} B(q,o) = 1 \quad \forall q \in S \tag{2}$$

Adjusting these parameters can be solved according to the idea of influential tutorial [26]. The work introduced an efficient framework which is a kind of iterative algorithm that is called Baum-Welch to estimate the parameters of an HMM. This algorithm is a special case of Expectation-Maximization (EM) algorithm that consists of two steps which are E-Step and M-Step. In E-Step, we count a variant of used transitions and observation symbols in the model. First, we calculate $\xi_t(q, q')$ which denotes the probability of being in state q at time t and state q' at time $t + 1$, given the observation sequence O and the current model θ:

$$\xi_t(q,q') = Pr(S_t = q, S_{t+1} = q'|O, \theta) \tag{3}$$

According to Bayes' theorem;

$$\xi_t(q,q') = \frac{\alpha_t(q)A(q,q')B(q',o_{t+1})\beta_{t+1}(q')}{Pr(O|\theta)} \tag{4}$$

where $q, q' \in S$ and $1 \leq t \leq T$ (*i.e.,* T is the length of observation sequence), $\alpha_t(q)$ is forward value, and the last $\beta_t(q)$ is backward calculation. Second, we calculate $\gamma_t(q)$ which determines the probability of being in hidden state q at time step t given the output sequence O and the model θ:

$$\gamma_t(q) = Pr(S_t = q|O, \theta) \tag{5}$$

If we apply Bayes' theorem then we rewrite the calculation as:

$$\gamma_t(q) = \frac{\alpha_t(q)\beta_t(q)}{Pr(O|\theta)} \tag{6}$$

In Equation 4 and Equation 6, there are two variables which are denoted as $\alpha_t(q)$ and $\beta_{t+1}(q')$ (or $\beta_t(q)$). Those are Forward and Backward calculations [5], respectively.

Forward-*Calculation:* It computes the likelihood of the observation by summing over the probabilities of all possible paths that led to the sequence of observations as shown in FIGURE 2.

The formal expression is:

$$\alpha_t(q) = Pr(o_1 o_2 \ldots o_t, S_t = q) \tag{7}$$

And the recursion formula is:

$$\alpha_t(q) = \sum_{i=1}^{N} \alpha_{t-1}(S_i)A(S_i,q)B(q,o_t) \tag{8}$$

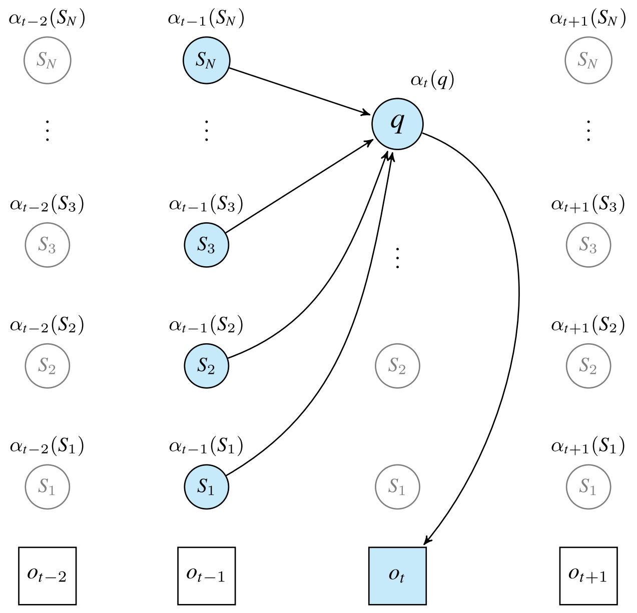


FIGURE 2. **The forward trellis: computing the total probability of the observation by summing all the previous forward values multiplied by transition probabilities and emission probability.**

where $q \in S$ and $1 \leq t \leq T$ and N is number of hidden state in the model. At the initial step, there is only one initial state (no initial vector π). In directly we calculate the forward value for each individual latent variable at the base step as

$$\alpha_1(q) = A(q_0, q)B(q, o_1) \quad q \in S, o_1 \in O \tag{9}$$

where q_0 denotes the initial state in the model.

Backward-*Calculation:* This calculation is the time-reversed version of the Forward calculation. It can be applied when we need to find the probability of observing all future events $o_{t+1:T}$ while being in state q at time step t as shown in FIGURE 3.

The conditional calculation is

$$\beta_t(q) = Pr(o_{t+1}o_{t+2}\dots o_T \mid S_t = q) \tag{10}$$

This calculation consists of the following steps:

- **Base Step**:

$$\beta_1(q) = 1 \quad \forall q \in S \tag{11}$$

- **Recursion Step**:

$$\beta_t(q) = \sum_{i=1}^{N} \beta_{t+1}(S_i)A(q, S_i)B(S_i, o_{t+1}) \tag{12}$$

where $q \in S$ and $1 \leq t \leq T$. In M-Step, the estimation of the state transition matrix needs two arguments which are the sum over all the number of the state transition from q to q', and the total number of transitions comes out from state q. The updated formula for transition matrix is

$$\hat{A}(q, q') = \frac{\sum_{t=1}^{T-1} \xi_t(q, q')}{\sum_{t=1}^{T-1}\sum_{k=1}^{N} \xi_t(q, S_k)} \quad S_k \in S \tag{13}$$

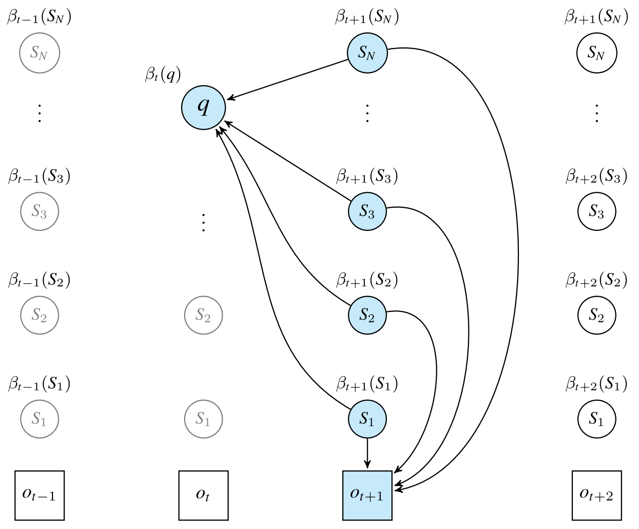


FIGURE 3. **The induction step of the *backward* procedure that proceeds by summing all the future $\beta_{t\mathcal{C}1}\triangleright S_i\triangleleft$ values with weighted state transition probability and future observation probability.**

The updated formula of the emission matrix also contains two arguments: sum of all time steps t of $\gamma_t(q)$ in which the observation symbol $o_t = o$ and sum of all $\gamma_t(q)$ for all time steps t.

$$\hat{B}(q, o) = \frac{\sum_{t=1}^{T} \mathbb{I}(o_t)\gamma_t(q)}{\sum_{t=1}^{T} \gamma_t(q)} \tag{14}$$

The notation $\mathbb{I}(o_t)$ represents that the corresponding observation symbol is o at time t where

$$\mathbb{I}(o_t) = \begin{cases} 1: & o_t = o \\ 0: & \text{otherwise.} \end{cases} \tag{15}$$

III. OBTAINING THE MOST LIKELY PATH IN STOCHASTIC HIDDEN INPUT AUTOMATA BY USING LIMITED OPTIMAL DISCRETE CONTROL

This section provides a modeling framework for calculating the most probable path in stochastic systems, similar to Hidden Markov Models (HMMs). Our approach, through symbolic discrete controller synthesis, offers a significantly more effective solution, both in terms of accuracy and under certain mutual constraints, compared to the widely used Viterbi algorithm for optimal path calculation.

In the modeling framework we propose, we consider the best path calculation in stochastic systems like HMMs as a problem of controlling discrete event systems. This theory is commonly used to synthesize a controller that accomplishes desired objectives in an uncontrolled system. Calculating the most probable path is viewed as an optimization goal, and our symbolic limited discrete controller synthesis algorithm aims to minimize the accumulated cost function over some time steps. As a result, the synthesized controller, by taking into account future states, facilitates state transitions in HMM to find the most probable path.

A. BACKGROUND OF THE DCS TECHNIQUE

The behavior of two discrete Mealy machines can be used to describe the control of discrete event systems. Mealy machines are mathematical models that are used to describe systems that produce outcomes in response to inputs. They are made up of a collection of states, inputs, outputs, and a transition function. The transition function accepts the current state and input and generates the next state and output.

When controlling discrete event systems, two Mealy machines are usually involved: one representing the system and one representing the supervisor. These Mealy machines represent an uncontrolled system that generates output based on input through state transitions. Similarly, a Mealy machine acting as a controller produces outputs that control system behaviors based on the inputs it receives from the system. Thus, the encapsulation variable enables the parallel synchronous operation of the controller's Mealy machines with desired behaviors.

The controller, created using relevant synthesis control algorithms, takes into account the system behaviors and satisfies the desired objectives. The desired objectives (eg., safety, optimization, reachability) are always guaranteed by means of synthesis control algorithms. The rules defined between inputs and outputs are subsequently utilized to construct a controller Mealy machine, also known as a DCS.

The resulting controller's reliability is guaranteed when the system behaviors are well-defined. However, the controller may not always guarantee the system's reliability if the system's actions are not well-defined. Therefore, it is essential to precisely and thoroughly represent the system's behaviors.

As a result, the utilization of Mealy machines and synthesis algorithms enables precise and reliable management of discrete event systems, ensuring that they perform as expected and meet the specified requirements.

The aim of this study is to control hidden systems, and the DCS tool, BZR, is used to do this utilizing the DCS principle. BZR offers a modeling environment, whose language is Heptagon. Heptagon is a parallel synchronous language, where the system and controller are represented as parallel compositions of data-flow equations, and the inputs and outputs are synchronized at discrete time steps. Heptagon is designed to express the parallel behavior of a system, making it suitable for modeling complex systems [1], [3], [14]. We apply appropriative synthesis algorithms for the constructed heptagon models by means of the BZR tool in order to generate controllers that ensure the system's behavior. We give details of the modeling framework with control means, below.

B. OVERVIEW OF THE MODELLING ALGORITHM FOR HIDDEN SYSTEMS

In this paper, we propose a systematic approach to obtain the most probable path for hidden systems, such as Hidden Markov Models (HMM). Our proposed approach focuses on obtaining the most likely path in such hidden systems.

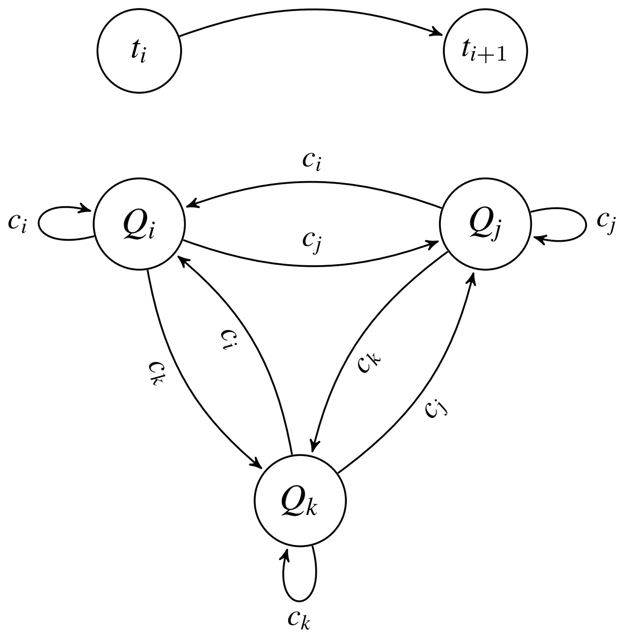


FIGURE 4. An example of 3 state explicit automaton modeled by using the HMM structure, where: Mealy machines symbolically encoded by the state in HMM and the state Q takes a value in $\{Q_i, Q_j, Q_k\}$ by means of the controllable variable c; and t denotes the time domain for state transitions.

To achieve this, we first symbolically encode a given HMM as a synchronous language, where the representation of HMM is replaced with a redecorated automaton as in FIGURE 4. The BZR environment, which can accept the parallel synchronous language, Heptagon, is used for this encoding. Then, using this encoding, we provide a framework for modeling hidden systems that includes control mechanisms for their analysis and synthesis. The details of this modeling framework are described below.

Our approach involves the following steps: (i) first, the behavior of a given HMM is transformed to a synchronous data-flow model $\mathcal{M}$, which is made of the synchronous parallel composition of "state" and "time", where the model is associated with control objectives; (ii) latter, safety and optimization objectives are identified, where: the safety objective supports the state transition in our model, and the optimization objective maximize the probability summed over a sliding window of a k-step of reactions of the system; (iii) last, the resulting controller for a given system with desired property, is computing by means of related algorithms, where the controller is a predicate involving "state" and "time". Then, we use this predicate on the script, which we design, by means of a traversing methodology to obtain the most probable path.

C. MODEL AND OBJECTIVES

List. 1 presents a generic structure of our synchronous data-flow model, denoted as $\mathcal{M}$, which is encoded using the language Heptagon. The $\mathcal{M}$encompasses both the given hidden system and the desired objectives. We describe a given HMM as a heptagon node. Each node consists of

```
 1 type Q_T = Q_1 | ... | Q_|Q|
   node 𝓜 () returns ()
   contract enforce c_act1 with c_1 , ... , c_|Q|: bool;
   var last Q: Q_T = Q_ss; last t: int = 0; p: int;
 5 let
     Q = if c_1 then Q_1 else ... if c_|Q| then Q_|Q| else last Q;
     t = last t + 1;
     c_act1 = (c_1 ⇒ ¬(c_2 or ... or c_|Q|)) or ... or
               (c_|Q| ⇒ ¬(c_1 or ... or c_|Q|−1));
10   p = if 𝒫_Q⊎t then A(q,q') * B(q',o) else 0;
   tel
```

LISTING 1. The generic structure of the HMM model.

sets of input and output variables, where the content of the model specifies that the input variables are determined by the output variables. The node declaration is provided in line 2, where the sets of input and output variables are enclosed in parentheses. In our case, we do not require any input/output transformations to model hidden systems. The body of the nodes is defined between the keywords "let" and "tel". These data-flows establish the valuation of the output and local flows by utilizing the instant valuation of states and inputs variables. States comprise memorized values represented with the "last" keyword. As indicated in line 3, our variables are defined using the "var" keyword. At each execution step, new values are assigned to the input flows, and the equations are collectively evaluated, resulting in the update of the output flow values accordingly. Line 1 introduces the enumeration type definition, resembling the syntax found in programming languages, utilizing the "type" keyword. The "contract" mechanism in line 3 enables us to specify our desired objectives and facilitates the synthesis of the controller.

We symbolically encode an automaton whose generic structure is given in FIGURE 4, where the automaton given in Section II is discretely described and decorated with controllable variables. We first identify a symbolic notation Q (at Line 4 in List. 1, where the initial value is starting state Q_{ss}) in the domain of enumerated type $Q_T = \{Q_1 \ldots Q_{|Q|}\}$ (at Line 1) for states in a HMM, where $|Q|$ denotes the number of states existing in the HMM. Q is encoded as a symbolic data-flow representation on the Heptagon node using the following equation:

$$Q = \begin{cases} Q_i, & if\ c_i \\ Q, & otherwise \end{cases} \tag{16}$$

State transition are represented over the value of Q by means of the controllable variable c_i[1] at Line 6 in List. 1, where the transition from Q_i to Q_j is occurred only by the true value of c_j at the synchronous firing[2] (at Line 7).

In addition, c_{act1} (at Line 8 in List. 1), which is an invariant (i.e., always "true"), supports to hold that only one input controllable variable c_i is active for a state transition at each time instant, defined as:

$$c_{act1} = \neg \bigoplus Q_i \backslash \bigwedge Q_j, \tag{17}$$

where $i, j \in |Q|, i \neq j$, and $\bigoplus$ denotes the operator EXOR.

The concept of the time domain involves modeling the processing of input data over time, where t can be considered either as a triggering event or as discrete time intervals. Within each interval, input data undergoes mathematical computations, including conditional statements, to determine the corresponding output data as time progresses, defined as:

$$t = t + 1 \tag{18}$$

We define the symbol p (represent that probability—at Line 9 in List. 1) as a cost function, where: the $\mathcal{P}_{Q\hat{a}t}$ is a predicate, involves current "state" and "time", in the domain of $\{true, false\}$; and the corresponding value of p is modeled as a natural number, which is the value of some ratio of given matrices A and B, defined as:

$$p = \begin{cases} A(q, q') * B(q', o), & if\ \mathcal{P}_{Q\hat{a}t} \\ 0, & otherwise \end{cases} \tag{19}$$

Last, we specify our safety objective c_{act1} as an invariant by means of the restriction of admissible values of controllable variables ($c_i, i \in |Q|$).

D. CONTROL MEANS

The algorithm for the proposed approach is outlined as follows: T is the length of the state sequence created in discrete time. k_{target} is the number of future target steps. sS denotes the state sequence that captures the best possible paths and is initialized with an initial state value at the beginning. The function $\mathcal{F}$ is a recursive function iterating over k and determining the state that provides the highest probability for each t. The function $\mathcal{F}$ takes three parameters: the iteration value k, the previous time step value, and the previous state value. The variable $\mathcal{P}$ holds the state that yields the highest probability at time t. The function max is responsible for selecting the state with the highest probability among those found within each same scope.

The DCS tool BZR [12] supports the contract mechanism by using the synchronous data-flow language Heptagon. Some of the other DCS tools, eg., Sigali and ReaX [3], similarly extend their environment with various additions to use in the compilation process. Contracts can be used to specify an *invariant* and *controllable flows* (which can take values as `true` and `false`) for the models.The compiler applies a synthesis algorithm for controller synthesis. The resulting supervisor is designed to impose constraints on the controllable flows, ensuring that the model complies with the given invariant condition. The enforcement of these constraints is accomplished by incorporating a contract within a heptagon node. The resulting supervisors are implemented as predicates for each controllable flow: the controller attempts to set c to $true$, unless doing so has the potential to violate the required invariant in future reactions.

[1]Each state value Q_i accompanied with a controllable input variable c_iâ$\{true, false\}$, where iâ$|Q|$.

[2]The symbol t refers to the time mentioned in Section II, and t synchronously triggers the transitions.

Algorithm 1 Pseudocode of Proposed Algorithm

```
1:  Initialization:
2:  $T = constant$
3:  $k_{target} = constant$
4:  $sS[T+1]$
5:  $sS[0] = initState$
6:
7:  Maximum Probability:
8:  for each step $t = 1, 2, 3, \ldots, T$ do
9:      sS[t]=$\mathcal{F}(0, t-1, sS[t-1])$
10: end for
11: procedure $\mathcal{F}(k, t, s)$
12:     if $(k = k_{target} \mid\mid t > T)$ then
13:         return 1
14:     end if
15:     $\mathcal{P} = 0$
16:     for each state $i$ do
17:         $\mathcal{P} = max(A(s,i) \cdot B(i,o) \cdot \mathcal{F}(k+1, t+1, i))$
18:     end for
19:     return $\mathcal{P}$
20: end procedure
21:
22: Path Backtracking:
23: for each step $t = 0, 1, 2, 3, \ldots, T$ do
24:     $s \leftarrow sS[t]$
25: end for
```

In our research, we have introduced a safety objective ϕ which ensures that only a state Q_i can be selected by its associated controllable variable c_i. This objective acts as a governing rule for these variables, ensuring that only one of them is active at a time, as specified in the safety objective. In addition, we have an optimization objective O which serves as another governing rule to maximize the probability value p using the k-step symbolic optimal control algorithm presented in [24].

Once the compilation process of BZR is complete, a controller is automatically synthesized using appropriate synthesis algorithms. This controller is designed to obtain the most likely string as in speech recognition based on the objectives and constraints specified in the safety and optimization objectives. We have conducted experimental evaluations to validate our approach, and the results are presented in the next section.

IV. EVALUATION

In our previous work [25], the DCS estimation was made for k=1. Nonetheless, in this work, we present an alternative answer for the issue of finding the most probable path in hidden frameworks by on one hand utilizing diverse k qualities and on the other hand, updating the model boundaries to notice its exhibition against Viterbi.

As it is known, in the Viterbi calculation, the likelihood of transition from a hidden state to other hidden states at each time slot is determined by multiplying the likelihood of the yield symbol emitted by the latent state in that time allotment, and its maximum value is kept. Despite the fact that DCS has a comparative estimation, ascertaining the future advances (this relies upon the k value) that the calculation presents to us simultaneously permits us to acquire a more proficient and more precise outcome.

As seen in FIGURE 5, two activities are acted in the Viterbi estimation. In the first place, the Viterbi esteems are determined up to the symbol of the last observation sequence, and afterward, the most probable path is found with the subsequent interaction called backtrace. Nonetheless, DCS does this with a single activity (where k=3 is given) and provides us with a similar outcome all the more successfully.

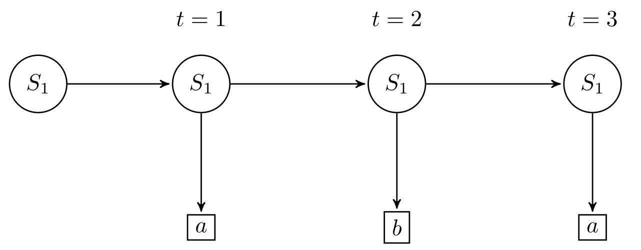
Initially, we encode HMM and its objectives as parallel synchronous dataflow equations in the ReaX. Subsequently, leveraging the synthesis algorithms offered by the DCS tool, ReaX compiler to obtain a controller. This controller will be used for the best path calculation in a simulation environment. Throughout these processes, the simulation environment is also implemented in Python using HMM. This integration ensures that the controller derived from the ReaX environment is seamlessly incorporated into the Python environment for the best path computation. Additionally, other methods are encoded and tested in the Python environment for comprehensive evaluation.

We made 100 random distinctive datasets of our own. These datasets contain various variations. For instance, we utilized distinctive sequence lengths and alternate quantities of state-symbol pairs as found in FIGURE 6. Here, the first esteem shows the hidden states utilized, while the other value shows the number of symbols used.

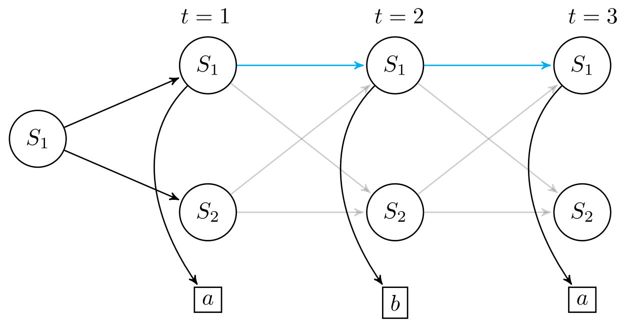
Then, at that point, we distinguished the factors for which each model (we likewise tried distinctive k qualities here) gave the most noteworthy exactness in the percentile (*i.e.,* sequence length, number of states, and symbols) and made a new dataset in which those factors were steady.

Then, we analysed the closeness of the decent models to the original model that made this dataset. At the point when we thought about the exactness densities, we observed that the fixed model, which is really settled with the expansion of the k value, gave results near the original model as sown in Figure 7.

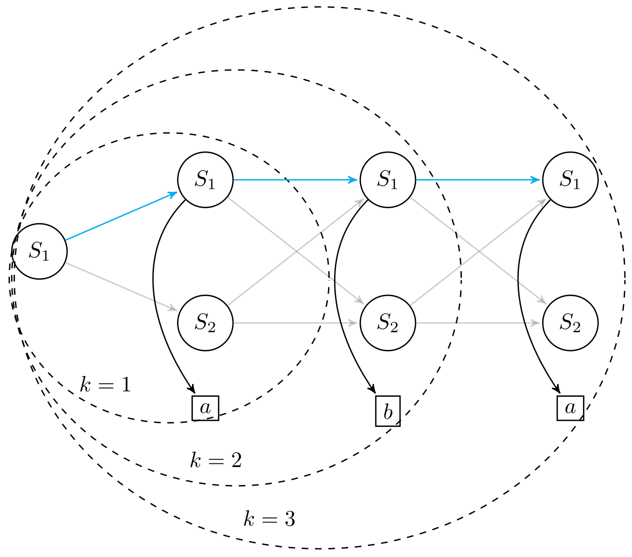
At long last, we analysed the error rate of each model utilizing the accuracy density we acquired in TABLE 1. As indicated by the qualities we got, we saw that there was a decline in the error rate with the increment of the k value. Nonetheless, when the value of k expands, the quantity of computations increments correlatively. In the current HMM, the computational complexity of the Viterbi algorithm is known to be quadratic. Our proposed algorithm, enhanced with the DCS technique, maintains the same Big-O notation when $k = 1$. However, as the values of k increase, the time required for computation and prediction will exponentially grow. The Big-O notation for our model is $O(TN^{k+1})$. To test this, we created different models with varying values of k and observed the computation times and the memory



(a) A 3 length output sequence generation by S_1,S_1,S_1



(b) Viterbi: the most probable sequence S_1,S_1,S_1 for the output sequence a, b, a



(c) DCS: the most probable path S_1,S_1,S_1 generates output sequence a, b, a for $k = 3$

FIGURE 5. (a) refers the current state sequence that generates the output a, b, a. (b) represent the Viterbi calculation that finds the most probable state sequence depending given output sequence a, b, a. (c) a DCS calculation considers the node which has high probability value is the state that also constitutes the output symbol.

usage, as shown in TABLE 2. It ought to be noticed that we haphazardly produced the raw datasets we utilized in the evaluation part. In any case, the precision upsides of the models were as we expected.

The assessment involved the computation of the Kullback-Leibler (KL) divergence in the following manner:

$$KL(P||Q) = P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{20}$$

TABLE 1. Error rate.

The Fixed Model	Error Calculation
Viterbi	%30.56
DCS (k=1)	%29.23
DCS (k=2)	%27.35
DCS (k=3)	%24.73

TABLE 2. Synthesis time and memory occupation.

k	Time (s)	Memory (KB)
1	0.22	55466
2	12.35	216094
3	62.08	591328
4	150.07	1760798
5	223.78	2784246
6	277.87	3132924
7	338.6	3333072

Let $Q(x)$ represent the probability matrices that are obtained during training for each variable x, and $P(x)$ represent the selected beginning probability matrices. The difference between these two likelihood distributions is measured by the KL score. A closer alignment between the distributions is shown by a lower KL score, and a precise correspondence is indicated by a score of zero. Therefore, the higher the model's precision, the closer the KL score is to zero.

When compared to previous studies, our proposed method demonstrated superior performance in the task of finding the most probable path in a stochastic system, as depicted in TABLE 3. Specifically, we extended the work in [25] by applying a k-step optimization using a synchronous data-flow model over a sliding time window. This allowed us to find the optimal path that maximizes the probability of the system's behavior while taking into account longer-term dependencies that are often present in real-world systems. In the subsequent step, we generated a new dataset with a model having 2 states and 5 observation symbols, based on the best KL score obtained. Each model in this dataset consisted of sequences of length 5000 and dimensions of 1000. We specifically compared this dataset using our proposed method against the Viterbi algorithm. The objective here was to observe whether the proposed method, as the value of k increased, would exhibit a more favorable KL score ratio compared to Viterbi, indicating better suitability. Additionally, we aimed to discern if there would be a convergence point beyond a certain k value where both methods yield the same score. As illustrated in TABLE 4, the anticipated convergence scenario was observed. Thus, despite the increase in k leading to an increase in computational complexity, it was observed that a certain optimal value of k could be reached.

Our methodology offers a number of advantages over existing approaches that have been utilized to tackle comparable issues, such as dynamic programming [4], reinforcement learning [33], and model predictive

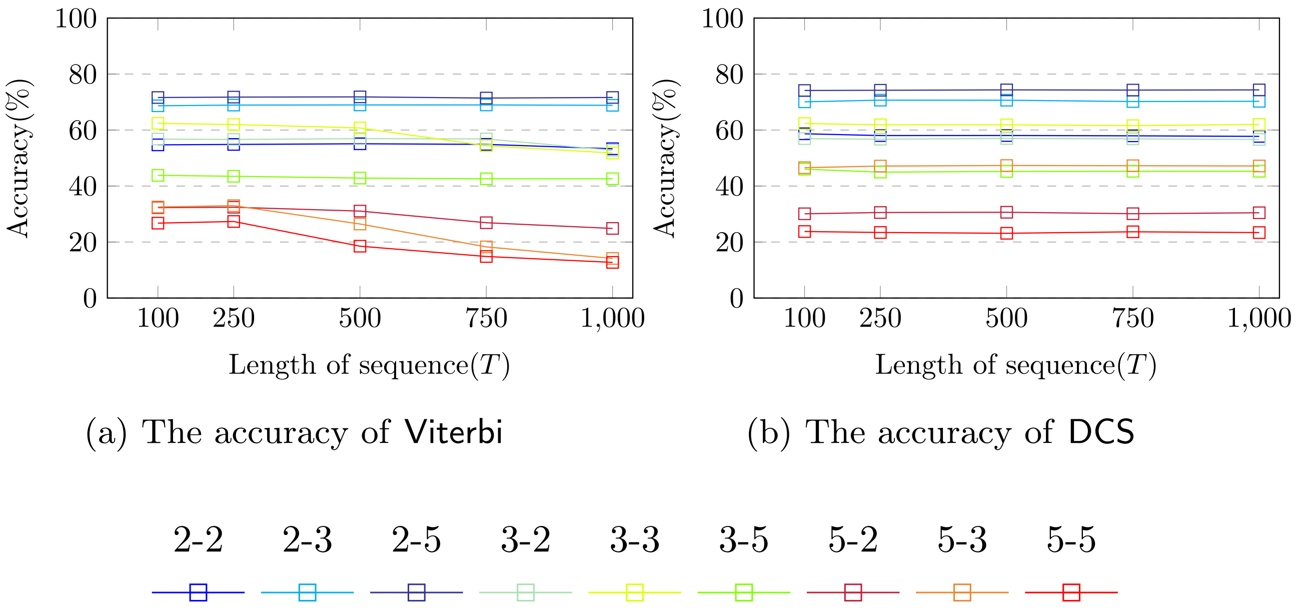


FIGURE 6. The accuracy diagram for different numerous of state-symbol pair.

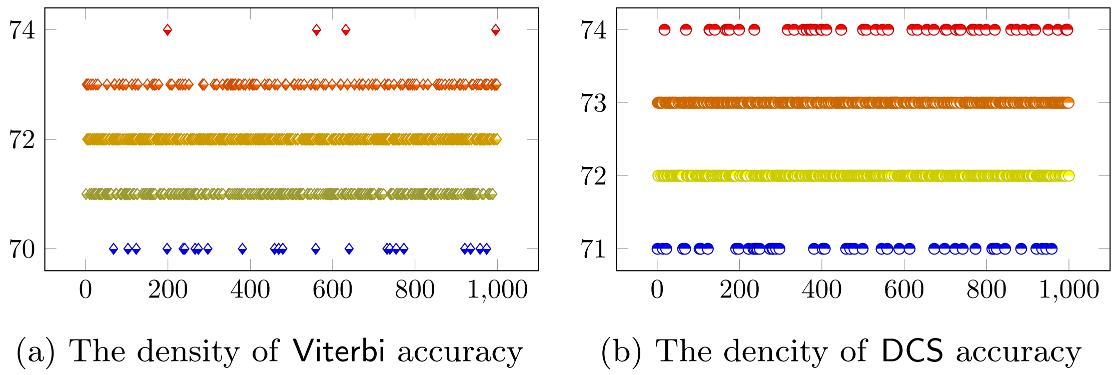


FIGURE 7. The density of accuracy both Viterbi and DCS according to the used dataset (2)-5 and 1000 sequences of 5000 each.

TABLE 3. KL divergence scores of SOTA methods.

Input	DCS	Viterbi	CYK	Adopted Viterbi [29]
S:2;O:2	0,1908	2,0159	44,6235	3,1576
S:2;O:3	0,0901	0,8409	27,67899	1,3213
S:2;O:5	0,0812	1,1896	23,8720	1,2287
S:3;O:2	0,1380	1,2633	29,3982	2,0152
S:3;O:3	0,1290	1,6178	39,0957	2,6538
S:3;O:5	0,3269	1,1928	41,5226	1.9869
S:5;O:2	0,3921	2,05843	32,7384	2,1343
S:5;O:3	0,2255	2,9588	51,2428	3,4756
S:5;O:5	1,2313	2,4217	63,6495	3.0018

control [28]. First off, our approach is founded on a formal framework for creating controllers that may accomplish a particular behavior while adhering to a set of requirements, as stated in [27]. This guarantees the system will behave in a safe and appropriate manner, which is essential in many real-life applications. In addition to determining the most likely path, our method can handle complex specifications, giving users more control over how the system behaves [34]. Moreover, our approach takes advantage of the problem's structure to condense the search space and boost convergence

TABLE 4. The comparison of KL scores for 2-5 model.

Method	KL Score
Viterbi	6.1425
DCS (k=1)	0.1942
DCS (k=2)	0.1339
DCS (k=3)	0.0855
DCS (k=4)	0.0793
DCS (k=5)	0.0028
DCS (k=6)	0.0028
DCS (k=7)	0.0028

rates, making it computationally effective and suitable for real-time applications [8].

The results showed that our approach is promising for the best path calculation for stochastic hidden systems. Without mentioning the destination state in the optimisation goals, our method takes the cost function into account. It consistently guarantees the specified system behaviours. For practical issues in reactive systems including probabilistic hidden systems, robotic systems, battery management systems, and traffic management systems, our approach shows promise.

V. CONCLUSION

Through the use of DCS we have provided a systematic methodology for determining the most likely path in a hidden stochastic system. The Viterbi algorithm is frequently used to determine the most likely course, however, it has some performance and reliability issues. Our method has been found to be more effective in terms of mathematical precision and reliability. The systematic framework we suggest allows us to use our approach to compute the most likely path in numerous hidden systems while also taking mutual constraints into account.

In our research, we treated the feedback control problem of hidden stochastic systems as a feedback stochastic system, and we modeled them using the BZR tool in the context of synchronous languages. When we applied our DCS algorithms to this model, we constructed the controller that would be used to decide the most likely course. The aims of our system were considered as a cost function by employing probability values. We then utilized our k-step optimal control strategy to maximize this cost function. We compared the performance of our method to the Viterbi algorithm using the updated parameters. Our findings demonstrated that by employing high-probability data, our model was more successful in locating optimal paths that were close to the true values.

Our approach is advantageous in many ways compared to many approaches found in the literature. The main advantage is that the approach we propose always guarantees system behavior with formal correctness. It can be easily modeled to address mutual constraints that cannot be managed in other approaches found in the literature, for multiple hidden stochastic systems. It allows for the use of cost-based optimization techniques without the necessity for a target state. With this versatility, such systems may more precisely model difficulties and produce better solutions to real-world challenges. Overall, our methodology has a number of advantages over previous methodologies, including guaranteed system behaviors, mutual constraint accommodation, better modeling, and precise responses for real-world challenges.

While our proposed method yields significantly better results in terms of accuracy, our approach is subject to certain constraints. The most significant limitation is the exponential increase in computational complexity with the increase in the k-step value. For instance, when our k value is equal to 1, our proposed model will have the same computational complexity as Viterbi. Our gain here lies in achieving better results than Viterbi. However, despite the increase in complexity with the rise of the k value, it also contributes to an enhancement in accuracy. Additionally, proportionally, the memory usage requirement increases with the augmentation of the k-step value in the DCS compiler.

As future work, our proposed approach can be adapted to various domains. For instance, in natural language processing, systems that construct using the most suitable tree structures often require high accuracy in predicting sentences. Our suggested approach can serve as an alternative method for researchers working in such domains, providing a novel perspective for achieving high accuracy in predictions. Furthermore, our approach can be easily applied to other stochastic systems besides HMM. For example, in stochastic counter models where accuracy values tend to be low, utilizing our approach for the best path calculation can prove highly beneficial, providing better accuracy.

In conclusion, our research proves the DCS method's efficiency in addressing the challenge of identifying the most likely course in a hidden stochastic system. Our method is particularly efficient in multiple hidden systems with mutual restrictions because it enables the synthesis of controllers that satisfy specified performance and safety requirements. The applicability of our method to additional optimization and scalability challenges, as well as its potential to solve other real-world issues, can also be researched.

REFERENCES

[1] X. An, "High level design control adapt. multiprocessor systems-on-chip," Ph.D. thesis, Dept. Comput. Sci., Université de Grenoble, Isere, France, 2013.
[2] C. Baier and J.-P. Katoen, "Discrete controller synthesis: A tutorial," *Theor. Comput. Sci.*, vol. 410, no. 49, pp. 4751–4769, 2009.
[3] N. Berthier and H. Marchand, "Discrete controller synthesis for infinite state systems with ReaX," *IFAC Proc. Volumes*, vol. 47, no. 2, pp. 46–53, 2014.
[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Chestnut, Nashua, USA: Athena Sci., 1995.
[5] C. M. Bishop, *Pattern Recognition and Machine Learning*, (Information Science and Statistics). Berlin, Germany: Springer-Verlag, 2006.
[6] B. Ma, Z. Liu, F. Jiang, W. Zhao, Q. Dang, X. Wang, J. Zhang, and L. Wang, "Reinforcement learning based UAV formation control in GPS-denied environment," *Chin. J. Aeronaut.*, vol. 36, no. 11, pp. 281–296, Nov. 2023.
[7] O. Cappe, E. Moulines, and T. Rydén, "Inference in hidden Markov models," in *Proc. EUSFLAT Conf.*, Lisbon, Portugal, Jan. 2005.

[8] N. Chatzipanagiotis and M. M. Zavlanos, "Distributed scheduling of network connectivity using mobile access point robots," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1333–1346, Dec. 2016.

[9] R. S. Chavan and G. S. Sable, "An overview of speech recognition using HMM," *Int. J. Comput. Sci. Mobile Comput.*, vol. 2, no. 6, pp. 233–238, Jun. 2013.

[10] M. Ciaperoni, A. Gionis, A. Katsamanis, and P. Karras, "SIEVE: A space-efficient algorithm for Viterbi decoding," in *Proc. Int. Conf. Manage. Data*, Jun. 2022, pp. 1136–1145.

[11] M. Cloosterman, L. Hetel, N. van de Wouw, W. Heemels, J. Daafouz, and H. Nijmeijer, "Controller synthesis for networked control systems," *Automatica*, vol. 46, no. 10, pp. 1584–1594, Oct. 2010.

[12] G. Delaval, E. Rutten, and H. Marchand, "Integrating discrete controller synthesis into a reactive programming language compiler," *Discrete Event Dyn. Syst.*, vol. 23, no. 4, pp. 385–418, Dec. 2013.

[13] Y. Di, R. Li, H. Tian, J. Guo, B. Shi, Z. Wang, K. Yan, and Y. Liu, "A maneuvering target tracking based on fastIMM-extended Viterbi algorithm," *Neural Comput. Appl.*, early access, pp. 1–10, Oct. 2023.

[14] E. Dumitrescu, A. Girault, H. Marchand, and É. Rutten, "Multicriteria optimal reconfiguration of fault-tolerant real-time tasks," *IFAC Proc. Volumes*, vol. 43, no. 12, pp. 356–363, Jan. 2010.

[15] Z. Ghahramani, *An Introduction to Hidden Markov Models and Bayesian Networks*. Singapore: World Scientific Publishing Co., Inc., 2001.

[16] S. Gong, M. Wang, B. Gu, W. Zhang, D. T. Hoang, and D. Niyato, "Bayesian optimization enhanced deep reinforcement learning for trajectory planning and network formation in multi-UAV networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 1–16, Mar. 2023.

[17] M. Kurucan, "Hidden probabilistic one-counter automata," PhD thesis, Univ. Liverpool, Anfield Rd, Anfield HQ, U.K., 2020.

[18] S. Liu, A. Trivedi, X. Yin, and M. Zamani, "Secure-by-construction synthesis of cyber-physical systems," *Annu. Rev. Control*, vol. 53, pp. 30–50, Jan. 2022.

[19] S. B. Liu, B. Schürmann, and M. Althoff, "Guarantees for real robotic systems: Unifying formal controller synthesis and reachset-conformant identification," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 1–15, Jun. 2023.

[20] B. Ma, Z. Liu, Q. Dang, W. Zhao, J. Wang, Y. Cheng, and Z. Yuan, "Deep reinforcement learning of UAV tracking control under wind disturbances environments," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[21] B. Ma, Z. Liu, W. Zhao, J. Yuan, H. Long, X. Wang, and Z. Yuan, "Target tracking control of UAV through deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 6, pp. 1–18, Mar. 2023.

[22] S. K. Monfared, O. Hajihassani, V. Mohsseni, D. Rahmati, and S. Gorgin, "A high-throughput parallel Viterbi algorithm via bitslicing," *ACM Trans. Parallel Comput.*, vol. 8, no. 4, pp. 1–25, Dec. 2021.

[23] N. Nguyen, "Hidden Markov model for stock trading," *Int. J. Financial Stud.*, vol. 6, no. 2, p. 36, Mar. 2018.

[24] M. Özbaltan and N. Berthier, "A case for symbolic limited optimal discrete control: Energy management in reactive data-flow circuits," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10688–10694, 2020.

[25] M. Ozbaltan and M. Kurucan, "Synchronous data-flow modeling for probabilistic hidden systems: An implementation of the most probable path calculation with 1-step symbolic optimal control algorithm," in *Proc. Int. Medit. Sci. Eng. Congr.*, 2021, pp. 120–125.

[26] L. R. Rabiner, *Readings in Speech Recognition*. Burlington, MA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 267–296.

[27] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete-event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[28] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*, San Francisco, CA, USA: Nob Hill Publishing, 2017.

[29] A. Sakharov and T. Sakharov, "The Viterbi algorithm for subsets of stochastic context-free languages," *Inf. Process. Lett.*, vol. 135, pp. 68–72, Jul. 2018.

[30] P. Schwab, G. C. Scebba, J. Zhang, M. Delai, and W. Karlen, "Beat by beat: Classifying cardiac arrhythmias with recurrent neural networks," in *Proc. Comput. Cardiol. (CinC)*, 2017, pp. 1–4, doi: 10.22489/CinC.2017.363-223.

[31] R. Schweiger, Y. Erlich, and S. Carmi, "FactorialHMM: Fast and exact inference in factorial hidden Markov models," *Bioinformatics*, vol. 35, no. 12, pp. 2162–2164, 2018.

[32] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3319–3331, May 2020.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[34] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*, Berlin, Germany: Springer Sci.+Bus. Media, 2009.

METE ÖZBALTAN received the Ph.D. degree in computer science from the University of Liverpool, in 2020. He is currently a Faculty Member with the Department of Computer Engineering, Erzurum Technical University.

MEHMET KURUCAN received the Ph.D. degree in computer science from the University of Liverpool, in 2020. He is currently a Faculty Member with the Department of Computer Engineering, Ardahan University.