

HIDDEN NON-MARKOVIAN MODELS: FORMALIZATION AND SOLUTION APPROACHES

C. Krull, G. Horton

Otto-von-Guericke-University Magdeburg, Germany

Corresponding author: C. Krull, Department of Simulation and Graphics, Otto-von-Guericke-University,
Universitätsplatz 2, 39106 Magdeburg, Germany, claudia@sim-md.de

Abstract. The simulation of real systems involves the building and analysis of models. Various types of discrete stochastic models (DSM) are well suitable to represent real processes with time dependent behavior, however to build a DSM, the system has to be completely observable. Hidden Markov Models (HMM) can model and analyze systems which are only observable through their interaction with the environment, but they are limited due to having a discrete-time Markov chain as hidden process. The combination of DSM and HMM will enable the utilization of the capabilities of HMM to non-Markovian models. The formalization of these so-called Hidden non-Markovian Models is presented in this paper. Furthermore the existing solution algorithms of HMM are adapted to the new modeling paradigm. However, this adaption was only successful for the evaluation and decoding algorithms and for models of Markov regenerative type, regenerating at every firing of a transition. The new modeling paradigm will be able to solve problems not solvable with existing methods of DSM or HMM alone. With fast and more generally applicable solution algorithms HnMM can be of use in many practical problems existing today.

1 Introduction

The modeling of real systems usually involves direct study and measurement of the systems' behavior. The observations are then simplified and idealized to build a model representing the relevant parts of the real system. However, not all systems can be observed directly. Some systems can only be examined through their interaction with the environment. These interactions can be interpreted as signals that the system emits and are often recorded in protocols. An example of such a system is a machine which produces either working or defective parts. It has two production modes, where the high performance mode is more efficient, but heats up the machine, so that it needs to be switched into low performance mode after a while. Depending on the production mode, the proportion of defective parts is different. The current mode of the machine is not transparent to the observer, but the condition of the parts produced can be determined. This analysis could for example enable a production manager to find out if the machine is working within the manufacturer's specification and when to schedule the next maintenance.

Using existing modeling paradigms, such protocols can not be used directly to build a simulation model. It is also not easily possible to use these signals and protocols to draw direct conclusions about the exact system behavior that produced them. Hidden Markov Models (HMM) can analyze hidden systems, but are limited in their modeling power. On the other hand, discrete stochastic models (DSM) have a large modeling power, but they have to be completely observable. Combining these two paradigms will enable the analysis of more realistic hidden models with the tools of HMM, specifically it will enable to link recorded system output to possible system behavior. This paper formalizes the combination of HMM and DSM to so-called Hidden non-Markovian Models. The formalization of the modeling paradigm is followed by an attempt to adapt some HMM to non-Markovian models. The presented algorithms can only be usefully implemented for non-Markovian models of Markov regenerative type, which regenerate at every firing of a transition. These HnMM will enable the solution of questions such as the most likely system behavior that produced a given output or the overall probability of producing an observed output sequence.

2 State of the Art

Hidden Markov Models are widely used in speech and pattern recognition [12]. They contain a discrete-time Markov chain (DTMC) as hidden model, which produces output signals in each discrete time step through a second stochastic process. Using HMM one can determine the absolute probability of a given signal sequence, which is called evaluation. Decoding enables to associate a given output signal sequence with the system behavior that produced it most likely. One can even train an unparameterized model to produce given desired output sequences. These tasks also require the definition of a sequence of output symbols (trace) representing distinct signals and a sequence of hidden system states (path) that might have produced the trace. All of these features are needed in speech and pattern recognition. However, the hidden model is a DTMC, which limits the modeling power of HMM to time independent state transitions rates, making the realistic representation of many real systems difficult.

Discrete stochastic models (DSM) are widely used in the simulation community for representing real systems behavior. There are several well known types of DSM, examples of these are the various forms of stochastic Petri

nets [2], stochastic reward nets [10] or stochastic activity networks [13]. The main advantage of DSM is that they enable the direct representation of processes and locations of the real system in the model. They also allow the use of time dependent transition rates, meaning, that processes can have any kind of distribution such as Weibull or Normal. In order to build a discrete stochastic model, one needs to be able to observe and measure the complete behavior of those parts of the real system that should be modeled. Using common simulation techniques such as discrete event-based simulation, it is not easily possible to determine the probability of an observed signal sequence or associate it with a specific system behavior that might have produced it.

Expanded Hidden Markov Models (eHMM) were introduced in [8]. These extend HMM by associating the signal output with the state changes, rather than with the discrete system states. This shifted the focus from system states to state changes, which are usually the elements of interest in a DSM. In [14] the Proxel-based simulation algorithm [6, 9] is used to solve the decoding and evaluation tasks for Hidden non-Markovian Models, without formally defining what HnMM are. In [7] an approach to train an HnMM using discrete phase-type distributions [11] and the original Baum-Welch algorithm was described.

The formal extension to discrete stochastic models as hidden model component is the next logical step. It will enhance the applicability of the new modeling paradigm and enable further research and improvement of the solution algorithms.

3 Formalization of Hidden non-Markovian Models

The combination of DSM and HMM requires the formal extension of the definition of expanded Hidden Markov Models. The first part is changing the hidden model part from a DTMC to a DSM. As the common ground of various types of discrete stochastic models, the definition is based on the state space of the model with transitions of arbitrary type between the states. It should be possible to build these state spaces from most DSM such as SPN [2], SAN [13] or SRN [10]. One could also define special types of HnMM for some classes DSM to ease modeling and enable the handling of unlimited state spaces.

Just like in eHMM, the stochastic symbol output process needs to be associated with the state changes, here the transitions of the model. Then the output symbol sequence and the internal state sequence need to be changed to match the new model definition. As a last step an adaption of the existing solution algorithms Forward, Viterbi and Baum-Welch [12] is attempted.

3.1 Special Cases of HnMM

Several special cases can be considered when designing an HnMM. Each of those special cases results from a decision between two options and might lead to a different formal description or modified solution algorithms.

The first decision is whether all state changes in the model emit symbols or not. The easier case is when all transitions produce output symbols (*Eall*). The more complicated, but also more realistic choice is that only some of the models transitions emit detectable symbols (*Esome*). *Eall* is easier to handle, because looking at an output sequence, each state change must have produced an output symbol, and the times of these state changes correspond exactly to the times points of the symbol emissions.

The second choice results from the decision to work on the state space of the model. It is possible when converting the DSM into its state space, that two transitions realize the same state change. The easier case is when each state change can only be realized by one transitions (*SConeT*). The more complicated model configuration is when multiply transitions can result in the change between the same two states (*SCnT*). *SConeT* is easier to handle, because there are less possibilities for hidden paths that the model development can take, resulting in less ambiguity.

The third choice is one specific to the class of non-Markovian models. When the transitions have other than Markovian firing time distributions, the activation time becomes important for the computation of the firing probability. One option is a Markov regenerative processes (MRP); Markov regenerative stochastic Petri nets are described in [3]. In an MRP the model is reset at specific points in time, so that the history before that point becomes irrelevant for the future development of the system, meaning that all non-Markovian transitions are reset and can be freshly activated, e.g. all age variables of non-Markovian transitions are set to 0. The easier case in HnMM is when all non-Markovian transitions are reset at every state change (*Treset*). Keeping the age of some transitions while other state changes happen (*Tkeep*) is the more realistic choice, but it is also mathematically more complicated to handle. *Treset* is easier to handle, because the discrete system state and the entry time, marking the activation time of the non-Markovian transitions, fully determine the systems future. When using *Tkeep*, one needs to remember the age of some non-Markovian transitions. This results in the need for supplementary variables [5]. The solution algorithms will then be very similar to the Proxel-based simulation method, which has already been used to solve HnMM tasks [14].

The three modeling choices that have to be made result in a number of possible combinations. To structure the following definitions and algorithm descriptions the easiest case (*Eall,SConeT,Treset*) will be considered the default.

We will then indicate the necessary changes for more complex cases.

3.2 Formal Description of an HnMM

The formalization of the HnMM definition is described in this section. The current formalization is based on [12], where a Hidden Markov Model is described as a 5-tuple (S, V, A, B, π) with the following components: a set of states S , a set of output symbols V , the transition probability matrix of the hidden DTMC A , a matrix describing the output process B and the initial probability vector of the DTMC π . The three elements (A, B, π) are also called the model, since they alone determine the actual dynamic behaviour of the HMM.

The sets S and V are kept in the definition of HnMM. The first addition to this definition for HnMM is a set of state changes C , so that an HnMM is described as a 6-tuple (S, C, V, A, B, π) . Since the state changes in a DSM can be of arbitrary type, the matrix A needs to reflect that. An equivalent to the state transition probability in the continuous non-Markovian case is the so-called instantaneous rate function (IRF) [6]. The IRF represents the continuous rate of flow of the random variable represented by the firing of the corresponding transition. It is defined as follows, where τ represents the age of the activated non-Markovian transition.

$$IRF(\tau) = \mu(\tau) = \frac{f(\tau)}{1 - F(\tau)}$$

The IRF of the exponential distribution is a constant. The elements of matrix A are the IRFs of the transitions between the discrete states of the model. Since these are in the general case dependent on the activation time of the corresponding transitions, matrix A is also time dependent. For the case of *SCnT*, where multiple transitions can realize the same state change, matrix $A(t)$ contains the sums of the transitions IRFs.

The stochastic output process of the model is described by B and can in general no longer be described as a matrix, when the symbol output is associated to the state changes. A state change is assigned a set of output symbols, each having an emission probability, the probabilities of the symbol emissions from one state change sum up to 1. For *Eall*, each transition is assigned the corresponding symbol output probabilities, where in *Esome* only the transitions producing detectable output need to be considered. Using *SConeT* the pair of states is sufficient to define the transition, *SCnT* makes it necessary to specify the exact transition, that induced a switch between two states.

The initial probability vector π containing the probabilities to start in each of the discrete model states can be used as is. The complete definition of an HnMM is the following:

$$\begin{aligned} HnMM &= (S, C, V, A, B, \pi) \\ Model : \lambda &= (A, B, \pi) \\ States : S &= \{s_1 \dots s_N\} \\ State Changes : C &= \{c_1 \dots c_L\} \quad (c_i \in S \times S) \\ Symbols : V &= \{v_1 \dots v_M\} \\ Transition Matrix : \\ A(t) &= \{a_{ij}(t)\}_{N \times N} \\ SConeT \ a_{ij}(t) &= \begin{cases} \mu_{ij}(t) & : \exists c_l = (s_i, s_j) \\ 0 & : else \end{cases} \\ SCnT \ a_{ij}(t) &= \sum_{c_l=(s_i, s_j)} \mu_l(t) \\ Output Probabilities : \\ SConeT \ B &= \{b_{ij}(m)\}_L \\ b_{ij}(m) &= P(v_m \text{ at } t_k | q(t_k - \varepsilon) = s_i \wedge q(t_k + \varepsilon) = s_j) \\ SCnT \ B &= \{b_i(m)\}_L \\ b_l(m) &= P(v_m \text{ at } t_k | (c_l, t_k)) \\ Initial Probability Vector : \pi &= \{\pi_i\}_N \\ \pi_i &= P(q(0) = s_i) \end{aligned}$$

The specification of Hidden Markov Models is closely related to the definition of a trace O (a sequences of observable output symbols) and a path Q (a sequence of hidden model states). These are necessary to define the tasks to be performed using HMM.

Changing the hidden model to a DSM also requires a change in the structure of the observable trace and the hidden path. The transitions of DSM can fire at arbitrary points in time. This requires every symbol representing a distinct output signal to be associated with the time stamp where it was produced. The trace becomes a sequence of pairs of the following form $(S, t) = (\text{symbol}, \text{timestamp})$. The transitions producing the output symbols are also associated with time stamps, turning the path into a sequence of pairs of the form $(T, t) = (\text{transition}, \text{timestamp})$. The path is no longer a state sequence, but now it is a sequence of state changes. The state sequence can easily be deduced from the sequence of state changes. The special case *SCnT* makes it necessary to specify the exact transition, that induced a switch between two states, otherwise with *SConeT* the pair of states is sufficient to define the transition. *Esome* makes the path possibly longer than the trace, the number of emitted symbols can be smaller than the number of state changes. This is possible, because state changes might have happened without emitting symbols.

$$\begin{aligned}
 \text{Symbol Sequence : } O &= \{(o_1, t_1) \dots (o_T, t_T)\} \\
 \text{State Change Sequence :} \\
 \text{Eall, SCnT } Q &= \{(c_1, t_1) \dots (c_T, t_T)\} (c_i \in C) \\
 \text{Eall, SConeT } Q &= \{((q_0, q_1), t_1), ((q_1, q_2), t_2) \dots ((q_{T-1}, q_T), t_T)\} (q_i \in S) \\
 \text{Esome, SCnT } Q &= \{(c_1, p_1) \dots (c_P, p_P)\} (c_i \in C, P \geq T) \\
 \text{Esome, SConeT } Q &= \{((q_0, q_1), p_1) \dots ((q_{P-1}, q_P), p_P)\} (q_i \in S, P \geq T)
 \end{aligned}$$

Having defined the structure of an HnMM, the following section describes how the existing solution algorithms for HMM can be adapted to HnMM.

4 Adapting Solution Algorithms for HnMM

Changing the model and sequence definition makes the existing algorithms for solving the HMM tasks obsolete. In this section we will attempt to adapt the original algorithms definitions taken from [12] to the definition of HnMM. This adaption of the algorithms is not always practical or even possible. Specific issues with the special cases are mentioned in the corresponding section. A following discussion summarizes the results and tries to asses, where and if algorithm adaption is possible and whether the results returned are still valid.

The key element of the following HnMM solution algorithms is the computation of the state change probability. For a time continuous model, the probability of a state change at a specific point in time t is 0. It is formally computed from the integral of the density function of the given distribution.

$$P(t \leq X \leq t) = \int_t^t f(x) dx = 0$$

Using this exact definition, the computation of trace probabilities would be futile. Therefore we need to determine the state change probability associated with a given symbol output in some other way. Using the modeling option *Eall*, here can be no hidden state changes between two symbol emissions. Then the path between two symbol emissions can only contain two states.

The simplest way to compute the state change probability for a certain point in time, is to use the probability to change the state between the activation of the transition and the actual emission of the symbol. This corresponds to the integral of the state change rate over the time elapsed since the last state change. For option *SConeT* this rate corresponds to the IRF of the one transition realizing that state change. The probability of the firing the transition (P_f) to induce the state change from i to j is the following, where t_k corresponds to the time of the last state change and t_{k+1} to the supposed firing time.

$$\begin{aligned}
 P_f(c_{ij}, t_k, t_{k+1}) &= \int_0^{t_{k+1}-t_k} a_{ij}(x) dx \\
 \text{SConeT} &= \int_0^{t_{k+1}-t_k} \mu_{ij}(x) dx
 \end{aligned}$$

When *Tkeep* is used, the activation time of the transition t_{act} and the time of the last state change t_1 might be different. Therefore, the estimation of the firing probability needs to be changed to reflect that.

$$\begin{aligned}
 P_f(c_{ij}, t_{act}, t_k, t_{k+1}) &= \int_{t_k - t_{act}}^{t_{k+1} - t_{act}} a_{ij}(x) dx \\
 SConeT &= \int_{t_k - t_{act}}^{t_{k+1} - t_{act}} \mu_{ij}(x) dx
 \end{aligned}$$

Another possibility to determine the state change probability is to consider the probability to remain in state i until time t_{k+1} and multiply that by a factor representing the fact to change state using the given transition. The probability to remain in a state can be computed by subtracting from 1 the probability to leave the state in the given time interval using any activated transition.

$$\begin{aligned}
 P_f(c_{ij}, t_k, t_{k+1}) &= \left(1 - \sum_m \int_0^{t_{k+1} - t_k} a_{im}(x) dx \right) * p \\
 Tkeep &= \left(1 - \sum_m \int_{t_k - t_{act}(c_{im})}^{t_{k+1} - t_{act}(c_{im})} a_{im}(x) dx \right) * p
 \end{aligned}$$

For option *Esome* a path between two symbol emissions can contain multiple state changes. Here the probability needs to be computed as the sum of all possible paths, which can become complex depending on the model specification.

The estimates of the state transition probability need to be tested experimentally and complemented by further alternatives. However, the determination of the state transition probability is also a matter of algorithm implementation. This implementation and related issues of information storage and retrieval are subjects of future work that are beyond the scope of this paper. For sake of simplicity, the first estimate using the integral of the IRF will be used as a place holder for the computation of the state change probability in the following algorithm definitions.

4.1 HnMM Evaluation - Finding the Probability of a Trace

The evaluation task has a given sequence of output symbols and a fully specified model. The task is to determine the probability with which the current model can emit the given output sequence, regardless of the internal behavior that produced it.

$$\text{Evaluation} : O, \lambda \Rightarrow P(O|\lambda)$$

The Forward Algorithm solves the evaluation problem by iteratively computing the probability of a given trace. The algorithm needs to be modified as follows to work for the simplest class of HnMM. One element $\alpha_{t_k}(i)$ is now the conditional probability to change to state i at time point t_k given the current model and trace up to t_k . Furthermore, the state change probability is now dependent on the time gap between two symbol emissions. The general algorithm structure does not need to be changed. the initialization and termination of the inductive algorithm can also be kept. An $\alpha_{t_{k+1}}(j)$ is computed as the sum of all possible predecessors $\alpha_{t_k}(i)$ multiplied by the path probability between the states and the probability to emit the given symbol o_{k+1} using the last state change.

SCnT makes the integration over the sum of the possible state changes necessary to determine the state change probability. *Esome* makes it possible to have multiple state changes between two symbol emissions, making it necessary to sum all possible paths from states i to j . The detailed implementation of this step is however not clear, since the possible number of state changes between two symbol emissions is dependent on the model definition and might be very large. *Tkeep* enables the activation and aging of transition independent of other state changes. Using this option, the relevant transitions ages need to be remembered, which will make the algorithm very similar to evaluation using Proxel-based simulation. The Forward algorithm for HnMM is the following:

$$\begin{aligned}
 \alpha_{t_k}(i) &= P((o_1, t_1) \dots (o_k, t_k), q(t_k + \varepsilon) = s_i | \lambda) \\
 &= P((o_1, t_1) \dots (o_k, t_k), ((*, s_i), t_k) \in \mathcal{Q} | \lambda)
 \end{aligned}$$

Initialization :

$$\alpha_0(i) = \pi_i$$

Induction : *SConeT, Eall, Treset*

$$\alpha_{t_{k+1}}(j) = \sum_{i=1}^N \left[\alpha_{t_k}(i) \int_0^{t_{k+1} - t_k} a_{ij}(x) dx b_{ij}(o_{k+1}) \right] \quad k = 0 \dots T - 1$$

$$\begin{aligned}
 SCnT &= \sum_{i=1}^N \left[\alpha_{t_k}(i) \sum_{c_l=(s_i, s_j)} \left[\int_0^{t_{k+1}-t_k} \mu_{c_l}(x) dx b_l(o_{k+1}) \right] \right] \quad k = 0 \dots T-1 \\
 SCnT, Esome &= \sum_{i=1}^N \left[\alpha_{t_k}(i) \sum_{\text{paths from } i \text{ to } j} \left[P(q(t_k) = s_i \wedge q(t_{k+1}) = s_j) \sum_{c_l=(*, s_j)} b_l(o_{k+1}) \right] \right] \\
 Tkeep &= \sum_{c_l=(s_i, s_j), i=1}^N \left[\alpha_{t_k}(i) \int_{t_k-t_{act}(c_l)}^{t_{k+1}-t_{act}(c_l)} a_{ij}(x) dx b_{ij}(o_{k+1}) \right] \quad k = 0 \dots T-1 \\
 \text{Termination :} \\
 P(O|\lambda) &= \sum_{i=1}^N \alpha_T(i)
 \end{aligned}$$

The Backward Algorithm also solves the evaluation problem by computing the overall probability of a trace. One element $\beta_k(i)$ holds the conditional probability to start in state i at time point t_k given the current model and trace from t_{k+1} to t_T . These elements are also needed for the Forward-Backward-Algorithm (see Section 4.3). The changes to the algorithm and modifications for the options are the same as in the Forward Algorithm.

$$\begin{aligned}
 \beta_{t_k}(i) &= P((o_{k+1}, t_{k+1}) \dots (o_T, t_T) | q(t_k + \varepsilon) = s_i, \lambda) \\
 &= P((o_{k+1}, t_{k+1}) \dots (o_T, t_T) | (*, s_i), t_k) \in Q, \lambda) \\
 \text{Initialization :} \\
 \beta_T(i) &= 1 \\
 \text{Induction :} & \quad SConeT, Eall, Treset \\
 \beta_{t_k}(i) &= \sum_{j=1}^N \left[\int_0^{t_{k+1}-t_k} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j) \right] \quad k = T-1 \dots 0 \\
 SCnT &= \sum_{j=1}^N \left[\sum_{c_l=(s_i, s_j)} \left[\int_0^{t_{k+1}-t_k} \mu_{c_l}(x) dx b_l(o_{k+1}) \right] \beta_{t_{k+1}}(j) \right] \quad k = T-1 \dots 0 \\
 SCnT, Esome &= \sum_{j=1}^N \left[\sum_{\text{paths from } i \text{ to } j} \left[P(q(t_k + \varepsilon) = s_i \wedge q(t_{k+1} + \varepsilon) = s_j) \sum_{c_l=(*, s_j)} b_l(o_{k+1}) \right] \beta_{t_{k+1}}(j) \right] \\
 Tkeep &= \sum_{c_l=(s_i, s_j), j=1}^N \left[\int_{t_k-t_{act}(c_l)}^{t_{k+1}-t_{act}(c_l)} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j) \right] \quad k = T-1 \dots 0 \\
 \text{Termination :} \\
 P(O|\lambda) &= \sum_{i=1}^N \beta_0(i) \pi_i
 \end{aligned}$$

4.2 HnMM Decoding - Finding the Most Likely Generating State Sequence

In this second problem, again the fully specified model and an output sequence are given. The task is to find the sequence of hidden states of the given model, which produced this trace most likely.

$$\text{Decoding : } O, \lambda \Rightarrow Q = \operatorname{argmax}_Q P(O|Q \wedge \lambda)$$

The Viterbi Algorithm is a greedy algorithm to determine the most likely valid generating path of a given trace [4]. The current adaption is only done for the option *Eall*. How to implement *Esome* is still unclear, because state changes not emitting symbols can only be guessed. The basic algorithm changes are again similar to the changes that were done in the Forward algorithm. One element $\delta_k(i)$ is now the probability of the most likely path so far that ends with the state change to i at time point t_k given the current model and trace up to t_k . The terms $\delta_{t_{k+1}}(j)$ is the maximum of all possible predecessors $\delta_{t_k}(i)$ multiplied by the path probability between the states and the probability to emit the given symbol o_{k+1} using the last state change.

The changes for *SCnT* and *Tkeep* are the same as for the Forward algorithm. For the option *Treset* the result of the Viterbi algorithm should be the most likely generating path. For *Tkeep* the greedy approach will most often not result in the most probable generating path, because the state change probability depends on a transitions age,

which changes over several symbol emissions, but the Viterbi algorithm only considers two adjoining symbol emissions in each computation. The complete formal adaption of the Viterbi algorithm to HnMM is the following:

$$\begin{aligned} \delta_k(i) &= \max_{q_0 \dots q_{k-1}} P(q_0 \dots q_{k-1}, q(t_k + \varepsilon) = s_i, (o_1, t_1)(o_2, t_2) \dots (o_k, t_k) | \lambda) \\ \delta_{t_{k+1}}(j) &= \max_{c_l=(s_i, s_j)} [\delta_k(i) * \text{state change probability} * b_l(o_{k+1})] \\ \text{Initialization :} & \\ \delta_0(i) &= \pi_i \\ \psi_0(i) &= 0 \\ \text{Recursion :} & \quad \text{SConeT, Eall, Treset} \\ \delta_k(j) &= \max_i \left[\delta_{t_{k-1}}(i) \int_0^{t_{k+1}-t_k} a_{ij}(x) dx b_{ij}(o_k) \right] \quad k = 1 \dots T \\ \text{SCnT} &= \max_{c_l=(s_i, s_j)} \left[\delta_{t_{k-1}}(i) \int_0^{t_{k+1}-t_k} \mu_{c_l}(x) dx b_l(o_k) \right] \quad k = 1 \dots T \\ \text{Tkeep} &= \max_i \left[\delta_{t_{k-1}}(i) \int_{t_{k-1}-t_{act}(c_l)}^{t_k-t_{act}(c_l)} a_{ij}(x) dx b_{ij}(o_k) \right] \quad k = 1 \dots T \\ & \quad \text{SConeT, Eall, Treset} \\ \psi_t(j) &= \operatorname{argmax}_i \left[\delta_{t_{k-1}}(i) \int_0^{t_{k+1}-t_k} a_{ij}(x) dx b_{ij}(o_k) \right] \quad k = 1 \dots T \\ \text{SCnT} &= \operatorname{argmax}_i \left[\delta_{t_{k-1}}(i) \int_0^{t_{k+1}-t_k} \mu_{c_l}(x) dx b_l(o_k) \right] \quad k = 1 \dots T \\ \text{Tkeep} &= \operatorname{argmax}_i \left[\delta_{t_{k-1}}(i) \int_{t_{k-1}-t_{act}(c_l)}^{t_k-t_{act}(c_l)} a_{ij}(x) dx b_{ij}(o_k) \right] \quad k = 1 \dots T \\ \text{Termination :} & \\ P^* &= \max_{i=1 \dots N} [\delta_T(i)] \\ q_T^* &= \operatorname{argmax}_{i=1 \dots N} [\delta_T(i)] \\ \text{Backtracking :} & \\ q_t^* &= \psi_{t+1}(q_{t+1}^*) \end{aligned}$$

4.3 HnMM Training - Finding the Model Specification for a Given Output

In this problem the starting point is an output symbol sequence. One also needs an initially parameterized model to start the training. The task is to find the parameterization of the model that best explains the output sequence. The initial model already somewhat predetermines the result, but the training task an optimization task than a direct computation.

$$\text{Training : } O \Rightarrow \lambda = \operatorname{argmax}_{\lambda} P(O|\lambda) \quad (1)$$

The Baum-Welch Algorithm is usually used to solve the training problem [1]. Training a model is by far the most complex of the three tasks of HMM. It will most likely not be directly portable to HnMM. One reason is that the training of an HMM also involves the modification of the state change probabilities of the DTMC. It is not clear how the IRFs of non-Markovian transitions can be modified in a meaningful way. The data used to modify the transition rates contains information only about certain parts of these functions. It needs to be tested, whether these modifications can be implemented still retaining a valid instantaneous rate function.

However, a direct adaption for the option *Eall* was attempted, even though it is unclear how to modify the elements of *A* to fulfill the training criteria. An adaption for *ESome* was not possible, because it is unclear which state transitions to train, when only some of the can be detected directly. Therefore no meaningful algorithm specification could be found yet. Using the option *SConeT*, the state changes can be trained. When using option *SCnT*, the individual transitions have to be trained.

If the modification of the a_{ij} (*SConeT*) and a_l (*SCnT*) is clear, one training epoch of a model will result in a larger probability to generate the given trace, but the optimality of the final resulting model specification can not be guaranteed. However, this is also true for the original Baum-Welch algorithm, where the result is also strongly dependent on the initial model parameterization.

One element $\xi_{t_k}(i, j)$ is the probability of the state change from i to j at time t_k given the output sequence and the current parameterization model. They can be computed from the terms $\alpha_{t_k}(i)$ and $\beta_{t_{k+1}}(j)$ taken from the Forward and Backward algorithms.

Baum-Welch Algorithm Using Option *SConeT*

$$\begin{aligned}
 \xi_{t_k}(i, j) &= P(q(t_k + \varepsilon) = s_i, q(t_{k+1} + \varepsilon) = s_j | O, \lambda) \\
 &= P(q(t_k - \varepsilon) = s_i, q(t_k + \varepsilon) = s_j | O, \lambda) \\
 T_{reset} &= \frac{\alpha_{t_k}(i) \int_0^{t_{k+1} - t_k} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j)}{P(O|\lambda)} \\
 &= \frac{\alpha_{t_k}(i) \int_0^{t_{k+1} - t_k} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j)}{\sum_{i=1}^N \sum_{j=1}^N \left[\alpha_{t_k}(i) \int_0^{t_{k+1} - t_k} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j) \right]} \\
 T_{keep} &= \frac{\alpha_{t_k}(i) \int_{t_k - t_{act}(s_i, s_j)}^{t_{k+1} - t_{act}(s_i, s_j)} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j)}{P(O|\lambda)} \\
 &= \frac{\alpha_{t_k}(i) \int_{t_k - t_{act}(s_i, s_j)}^{t_{k+1} - t_{act}(s_i, s_j)} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j)}{\sum_{i=1}^N \sum_{j=1}^N \left[\alpha_{t_k}(i) \int_{t_k - t_{act}(s_i, s_j)}^{t_{k+1} - t_{act}(s_i, s_j)} a_{ij}(x) dx b_{ij}(o_{k+1}) \beta_{t_{k+1}}(j) \right]} \\
 \gamma_k(i) &= \sum_{j=1}^N \xi_{t_k}(i, j) \\
 \sum_{k=1}^T \gamma_k(i) &= \text{expected number of transitions from } s_i \\
 \sum_{k=1}^T \xi_{t_k}(i, j) &= \text{expected number of transitions from } s_i \text{ to } s_j \\
 \text{Reestimation :} \\
 \bar{\pi}_i &= \text{expected frequency in state } s_i \text{ at time } t = 0 \\
 &= \gamma_1(i) \\
 \bar{a}_{ij} &= \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of transitions from } s_i} \\
 &= \frac{\sum_{k=1}^T \xi_{t_k}(i, j)}{\sum_{k=1}^T \gamma_k(i)} \\
 \bar{b}_{ij}(m) &= \frac{\text{expected number of transitions from } s_i \text{ to } s_j \text{ observing symbol } v_m}{\text{expected number of transitions from } s_i \text{ to } s_j} \\
 &= \frac{\sum_{k=1}^T \sum_{o_k = v_m} \xi_{t_k}(i, j)}{\sum_{k=1}^T \xi_{t_k}(i, j)}
 \end{aligned}$$

Baum-Welch Algorithm Using Option *SCnT*

$$\begin{aligned}
 \xi_{t_k}(l) &= P(c_l = (s_i, s_j) \text{ happens at } t_k | O, \lambda) \\
 &= P(q(t_k - \varepsilon) = s_i, q(t_k + \varepsilon) = s_j | O, \lambda) \\
 T_{reset} &= \frac{\alpha_{t_k}(i) \int_0^{t_{k+1} - t_k} \mu_l(x) dx b_l(o_{k+1}) \beta_{t_{k+1}}(j)}{P(O|\lambda)} \\
 &= \frac{\alpha_{t_k}(i) \int_0^{t_{k+1} - t_k} \mu_l(x) dx b_l(o_{k+1}) \beta_{t_{k+1}}(j)}{\sum_{i=1}^N \sum_{j=1}^N \left[\alpha_{t_k}(i) \sum_{c_l=(s_i, s_j)} \left[\int_0^{t_{k+1} - t_k} \mu_l(x) dx b_l(o_{k+1}) \right] \beta_{t_{k+1}}(j) \right]} \\
 T_{keep} &= \frac{\alpha_{t_k}(i) \int_{t_k - t_{act}(c_l)}^{t_{k+1} - t_{act}(c_l)} \mu_l(x) dx b_l(o_{k+1}) \beta_{t_{k+1}}(j)}{P(O|\lambda)} \\
 &= \frac{\alpha_{t_k}(i) \int_{t_k - t_{act}(c_l)}^{t_{k+1} - t_{act}(c_l)} \mu_l(x) dx b_l(o_{k+1}) \beta_{t_{k+1}}(j)}{\sum_{i=1}^N \sum_{j=1}^N \left[\alpha_{t_k}(i) \sum_{c_l=(s_i, s_j)} \left[\int_{t_k - t_{act}(c_l)}^{t_{k+1} - t_{act}(c_l)} \mu_l(x) dx b_l(o_{k+1}) \right] \beta_{t_{k+1}}(j) \right]} \\
 \gamma_k(i) &= \sum_{c_l=(i,*)} \xi_{t_k}(l)
 \end{aligned}$$

$$\sum_{k=1}^T \gamma_k(i) = \text{expected number of transitions from } s_i$$

$$\sum_{k=1}^T \xi_k(l) = \text{expected number of transitions through } c_l = (s_i, s_j)$$

Reestimation :

$$\begin{aligned} \bar{\pi}_i &= \text{expected frequency in state } s_i \text{ at time } t = 0 \\ &= \gamma_1(i) \\ \bar{a}_l &= \frac{\text{expected number of transitions through } c_l = (s_i, s_j)}{\text{expected number of transitions from } s_i} \\ &= \frac{\sum_{k=1}^T \xi_k(l)}{\sum_{k=1}^T \gamma_k(i)} \\ \bar{b}_l(m) &= \frac{\text{expected number of transitions through } c_l = (s_i, s_j) \text{ observing symbol } v_m}{\text{expected number of transitions through } c_l = (s_i, s_j)} \\ &= \frac{\sum_{k=1}^T \xi_k(l) \delta_{o_k=v_m}}{\sum_{k=1}^T \xi_k(l)} \end{aligned}$$

The adaption of the algorithms described in this section are subject to a number of restrictions, mostly due to the options chosen for the model. Some of the algorithms can be implemented directly, others contains parts that still need to be specified. The following section will discuss the possibilities and limitations of the HnMM formalization and algorithm adaptations.

5 Discussion of HnMM Definition and Solution Algorithms

The previous sections described the new definition of HnMM and some modified algorithms to solve the three tasks evaluation, decoding and training for non-Markovian models. The Forward and Viterbi algorithms could easily be adapted for the most simple HnMM configuration (*Eall, SConeT, Treset*). Here every model transition emits detectable output symbols, one state change is realized by only one transition and the modeled process is of Markov regenerative type, meaning, that all non-Markovian transitions are reset at the firing of any model transition. The two algorithms could also be adapted to the option *SCnT*. The Forward algorithm could theoretically handle *Esome*, where only some of the model transitions emit symbols.

The adaption to *Tkeep*, where the process is truly non-Markovian, was attempted, but can not be considered a success. When concurrently activated non-Markovian transitions are not reset, but keep their activation time, then these activation times need to be remembered as transition ages. These ages are not part of the discrete model states, but expand them possibly leading to a state space explosion. The algorithms only consider the discrete model states. To be correct, the states expanded by the ages need to be considered separately. As mentioned above, this eventually results in the method of supplementary variables [5] which has already been applied to HnMM [14].

Furthermore, it is currently not obvious how to implement the adaption of the Baum-Welch algorithm to HnMM. This was however expected, because training non-Markovian transitions via their instantaneous rate functions and only based on data concerning certain parts of the functions does not seem feasible. If this problem can be solved or if the problem is not as acute, as it appears, then the Baum-Welch can also be used to train HnMM of type *Eall, Treset* and for both options *Eall* and *Esome*.

Considering all of these restrictions, the adaption of the algorithms was only partially successful. However, the main contribution of the paper consists in the formalization of HnMM themselves. In previous papers alternative solution algorithms have already been considered, so that the adaption of the algorithms was not necessary to make HnMM useful.

We hope that HnMM as a combination of DSM and HMM will combine the advantages of both paradigms. The new paradigm will enable the following: Protocols or other output sequences of real systems can be used to deduce possible system behavior that produced them. One can determine the probability of a certain output sequence. The training of HnMM can make it possible to model systems that have a general continuous time behavior which are not directly observable. In general, the extension of HMM to more general hidden models utilizes the capabilities of HMM for more realistic representations of real systems.

6 Example Hidden non-Markovian Model of a Machine

The example of the machine with two production modes that produces defective and working parts will be used to illustrate the modeling capabilities of HnMM. The machine has two production modes. In high performance

mode the machine produces a larger portion of working parts, but this mode also heats up the machine. Switching to low performance mode reduces the portion of working parts, but it also regulates the internal temperature. Due to terms of guarantee or knowledge protection policies the production manager is not allowed to open the machine for observation purposes. An observed sequence of the produced parts' condition is then the output of the hidden machines condition.

A graphical representation of the model is shown in Figure 1. The broken line represents the boundary of the not observable hidden model, which is only crossed by the symbols emitted by two transitions. The internal model state space can be represented by a stochastic Petri net, since the places of the SPN in this case correspond to the possible discrete model states. As can be seen, only the two transitions *HP_Produce* and *LP_Produce* emit symbols (produce parts), therefore the model is of type *Esome*. The transitions *Overheat* and *CoolDown* do not emit symbols, meaning that the switch between the two production modes can not be directly detected from the outside. The smaller portion of working parts is represented by a smaller probability to emit the symbol *WP - Working Part* when firing transition *LP_Produce*. Each state change is realized by exactly one transition, making the model of type *SConeT*. Whether the model is of type *Tkeep* or *Treset* depends on the type of the transitions switching between the two production modes. If the transitions are of type *race age*, then model is of type *Tkeep*. When the transitions are of type *race enable*, the model is of type *Treset*, regardless of the actual distribution function associated with the transition. Considering the stated problem of heating up the machine, *Tkeep* would make more sense here. If the model was built differently, so that the production of a part did not induce a state change, then any non-Markovian distribution of the switching transitions (*Overheat* and *CoolDown*) would result in a model of type *Tkeep*. However, the current definition of HnMM only allows transitions that actually result in a state change.

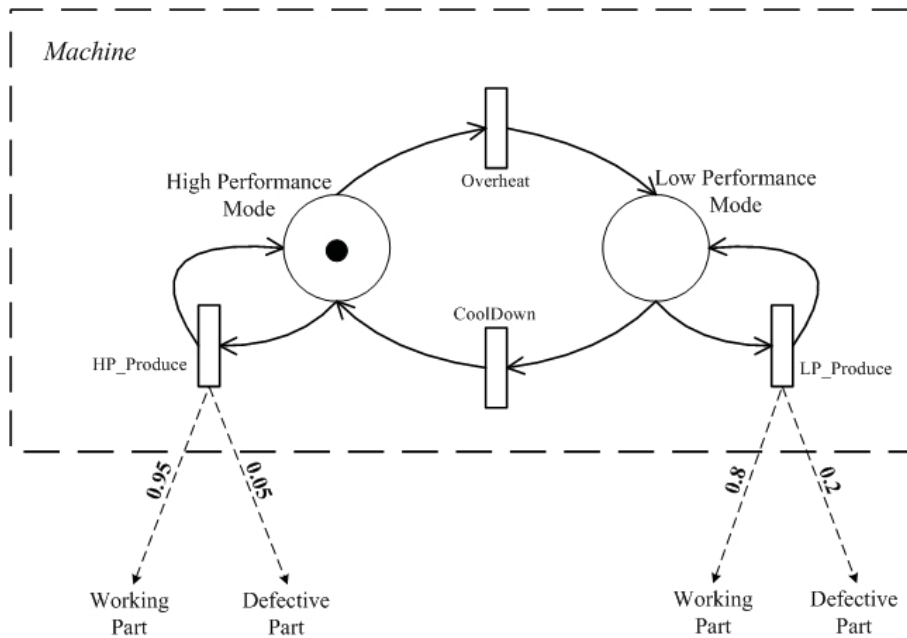


Figure 1: Example Hidden non-Markovian Model of Machine with two Production Modes

The formal specification of the example model can be found below. For sake of clarity the long names of the transitions and states are abbreviated.

$$\begin{aligned}
 \text{States : } S &= \{HighPerformanceMode, LowPerformanceMode\} \\
 &= \{HPP, LPP\} \\
 \text{State Changes : } C &= \{Overheat, CoolDown, LP_Produce, HP_Produce\} \\
 &= \{OH, CD, LPP, HPP\} \\
 \text{Symbols : } V &= \{WorkingPart, DefectivePart\} \\
 \text{Transition Matrix :} \\
 A(t) &= \begin{bmatrix} \mu_{HPP}(t) & \mu_{OH}(t) \\ \mu_{CD}(t) & \mu_{LPP}(t) \end{bmatrix} \\
 \text{Output Probabilities : } b_{HPP}(WP) &= 0.95 \\
 b_{HPP}(DP) &= 0.05 \\
 b_{LPP}(WP) &= 0.8
 \end{aligned}$$

$$b_{LPP}(DP) = 0.2$$

$$\text{Initial Probability Vector : } \pi = \{1, 0\}$$

A possible output symbols sequence and a possible generating path are the following:

$$\begin{aligned} \text{Symbol Sequence : } O &= \{(WP, 2.1), (WP, 4.2), (WP, 5.8), (WP, 8.0), (DP, 9.9), \dots\} \\ \text{State Change Sequence : } Q &= \{((HPM, HPM), 2.1), ((HPM, HPM), 4.2), ((HPM, HPM), 5.8), \\ &\quad ((HPM, LPM), 6.8), ((LPM, LPM), 8.0), ((LPM, LPM), 9.9), \dots\} \\ &= \{(HPP, 2.1), (HPP, 4.2), (HPP, 5.8), (OH, 6.8), (LPP, 8.0), (LPP, 9.9), \dots\} \end{aligned}$$

The evaluation problem applied to this example model can determine the probability of a specific observed sequence given the manufacturers specification of the machine. If this probability is lower than expected, then this can indicate a problem or the machine might not work within the manufacturer's specifications. The decoding task gives the most likely generating path of an observed sequence. Comparing this state sequence to the manufacturers specification, one could check whether the machine is switched to low mode in time not to overheat or if it works in high performance mode most of the time. This can help to decide if a scheduled maintenance should be done earlier or when the machine might need replacement. The solution to these tasks can be computed using the algorithms shown in this paper, if the model can be built in type *Treset*. Otherwise the Proxel-based simulation method needs to be used as described in [14].

7 Conclusion

The paper presents the formalization of HnMM, which are a combination of Hidden Markov Models and discrete stochastic models. The most important part is the formal definition of an HnMM and the identification of some special cases. The attempted adaption of the HMM solution algorithms was only partially successful. The training method is currently not applicable to HnMM and the evaluation and decoding tasks can only be solved if the model is of Markov regenerative type. However, other solutions for the training and more realistic models have already been described. An example of a HnMM was shown to illustrate the capabilities of the new modeling paradigm.

The new paradigm combines advantages of both HMM and DSM, and thereby makes it possible to solve problems not solvable with existing paradigms. The formalization of HnMM is a necessary step to the further examination and development. One drawback is that currently the existing and the adapted solution algorithms are computationally expensive. Increasing the efficiency of the algorithms is a task of future research. Another task is the comparison of the modified original HMM algorithms and the solution algorithms based on Proxels and phase-type distributions. We hope that in the future HnMM will present a practically applicable tool for analysts in various fields.

8 References

- [1] Baum, L. E. and Petrie, T. *Statistical inference for probabilistic functions of finite state markov chains*. The Annals of Mathematical Statistics, 37:1554–1563, 1966.
- [2] Bobbio, A., Puliafito, A., Telek, M., and Trivedi, K. S. *Recent developments in non-markovian stochastic petri nets*. Journal of Systems Circuits and Computers, 8(1):119–158, 1998.
- [3] Choi, H., Kulkarni, V. G., and Trivedi, K. S. *Markov regenerative stochastic petri nets*. Perform. Eval., 20(1-3):337–357, 1994.
- [4] Forney, G. D. *The viterbi algorithm*. In: Proceedings of the IEEE, vol. 61, 268–278. 1973.
- [5] German, R. *Transient analysis of deterministic and stochastic petri nets by the method of supplementary variables*. In: Proceedings of the 3rd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 95), 394–398. IEEE Computer Society, 1995.
- [6] Horton, G. *A new paradigm for the numerical simulation of stochastic petri nets with general firing times*. In: Proceedings of the European Simulation Symposium 2002, 129–136. SCS European Publishing House, 2002.
- [7] Isensee, C., Wickborn, F., and Horton, G. *Training hidden non-markov models*. In: Proceedings of 13th International Conference on ANALYTICAL and STOCHASTIC MODELLING TECHNIQUES and APPLICATIONS, Bonn, Germany, 105–110. 2006.
- [8] Krull, C. and Horton, G. *Expanded hidden markov models: Allowing symbol emissions at state changes*. In: Proceedings of 14th International Conference on ANALYTICAL and STOCHASTIC MODELLING TECHNIQUES and APPLICATIONS, Prague, Czech Republic, 185–190. 2007.

- [9] Lazarova-Molnar, S. *The proxel-based method: Formalisation, analysis and applications*. Ph.D. thesis, Otto-von-Guericke-University Magdeburg, 2005.
- [10] Muppala, J., Ciardo, G., and Trivedi, K. S. *Stochastic reward nets for reliability prediction*. *Communications in Reliability, Maintainability and Serviceability*, 1(2):9–20, 1994.
- [11] Neuts, M. F. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. The John Hopkins University Press, Baltimore, 1981.
- [12] Rabiner, L. R. *A tutorial on hidden markov models and selected applications in speech recognition*. In: *Proceedings of the IEEE*, vol. 77, 257–286. 1989.
- [13] Sanders, W. H. and Meyer, J. F. *Stochastic activity networks: formal definitions and concepts*. 315–343, 2002.
- [14] Wickborn, F., Isensee, C., Simon, T., Lazarova-Molnar, S., and Horton, G. *A new approach for computing conditional probabilities of general stochastic processes*. In: *Proceedings of 39th Annual Simulation Symposium 2006*, Huntsville, USA, 152–159. 2006.