Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

HIDDEN NON-MARKOVIAN REWARD MODELS : VIRTUAL STOCHASTIC SENSORS FOR HYBRID SYSTEMS

Claudia Krull
Graham Horton

Otto-von-Guericke-University
Universitaetsplatz 2
39106 Magdeburg, GERMANY

ABSTRACT

We are interested in partially observable hybrid systems whose discrete behavior is stochastic and unobservable, and for which samples of some of the continuous variables are available. Based on these samples of the continuous variables, we show how the hidden discrete behavior may be reconstructed computationally, which was previously not possible. The paper shows how Hidden non-Markovian Models (HnMM) can be augmented with arbitrary rate and impulse rewards to model these partially observable hybrid systems. An HnMM analysis method is adapted to find the probability of a sample sequence for a given model, as well as likely system behaviors that caused the observation. Experiments illustrate the analysis method and the possible complexity of the reward measure through a medical example and one from computer gaming. The paper extends the class of partially observable systems analyzable via virtual stochastic sensors into the continuous realm for the first time.

1 INTRODUCTION

Currently, partially observable systems are represented by unobservable discrete stochastic processes, which output symbols at discrete points in time. Real systems often contain continuous elements, which have to be discretized, loosing accuracy, or completely ignored. Virtual stochastic sensors (Krull, Buchholz, and Horton 2011) have been developed to reconstruct hidden stochastic behavior based on observable behavior of PODS systems. HnMM are used as computational model to represent the PODS systems. Different analysis methods for HnMM are available that can perform behavior reconstruction.

To include continuous behavior, discrete stochastic systems can be augmented with rewards to represent continuous measures such as performance or costs (SAN or SRN). These continuous measures can change according to the discrete system state. We propose to augment partially observable discrete stochastic (PODS) systems similarly to incorporate arbitrary continuous behavior in the form of rewards. We assume that the discrete state of our system is not observable, but we have samples of some continuous measures at discrete points in time. Our goal is to reconstruct the unobservable internal system behavior and / or complete reward development based only on these sequences of reward samples.

One example of such a system from computer gaming tries to determine an opponents strategy (discrete system) based on samples of the continuous opponents army size, corresponding to the HMM evaluation task. Another example is a diabetic, where the doctor tries to determine the actual eating and insulin injection times (discrete system) based on the measurements of the continuous blood glucose level, corresponding to the HMM decoding task. For these hybrid models behavior reconstruction is currently not possible using existing HnMM, these can only cope with discrete systems and output. However, the similarity between (impulse) rewards and symbol emissions suggests that HnMM and existing solution methods are adaptable to cope with continuous reward values.

Our idea is to adapt the existing HnMM specification to include rewards instead of symbol emissions, resulting in Hidden non-Markovian Reward Models (HnMRM). We will also adapt an existing reconstruction algorithm to interpret samples of continuous reward measure instead of symbol emissions. This new model class HnMRM and algorithms would enable analysis or identification of an underlying discrete stochastic process based on samples of a continuous reward measure. This increases the space of analyzable POS systems with the help of HnMM and extends virtual stochastic sensors into the realm of hybrid and continuous systems with stochastic influences.

The paper will first review the state of the art on partially observable systems and corresponding analysis methods. Then HnMRM are defined formally and the analysis approach sketched. Experiments show the validity and applicability of the analysis methods.

2 STATE OF THE ART

2.1 Virtual Stochastic Sensors and Hidden Non-Markovian Models

Virtual Stochastic Sensors enable the analysis of unobservable processes in discrete stochastic systems. Virtual sensors (Ibargüengoytia and Reyes 2006) use physical sensor readings to deduce the value of a quantity of interest. In virtual stochastic sensors (VSS), the physical sensor readings and their relationship with the quantity of interest are stochastic. Therefore the measurement of our VSS is a statistical estimate of the true value. Hidden non-Markovian Models are the mathematical model that enables the modeling of the stochastic relationships and the computation of the results of the VSS. (Krull, Buchholz, and Horton 2011)

Hidden Markov Models (HMM) (Fink 2008) form the formal basis of Hidden non-Markovian Models (HnMM). HMM are discrete-time Markov chains (DTMCs) (Bolch, Greiner, de Meer, and Trivedi 1998) that produce discrete output symbols based on the hidden DTMC state and given emission probabilities. HMM are memoryless and discrete, because they are based on DTMCs. Three basic tasks are defined for HMM: Evaluation determines the probability of a given trace for a given model, Decoding determines the most likely generating path for a given trace and model and Training iteratively improves an initial model to increase the probability of a given trace or set of traces (Fink 2008). Efficient algorithms exist for all three HMM tasks.

HnMM extend HMM by adding time dependency, stochastic firing times of transitions and time stamps to symbol emissions. Furthermore, symbol emissions can depend on the current system state (Krull, Buchholz, and Horton 2010) or on the state changes (Krull, Buchholz, and Horton 2011). HnMM are a computational model and can use an augmented stochastic Petri net (ASPN) (Buchholz 2012) as user model. These ASPN augment normal stochastic Petri nets as defined by Krull (2008) with output symbols at the state transitions. One particular marking of the ASPN can be transferred into one state of the computational model HnMM and one transition of the ASPN can result in multiple state changes of the HnMM. The complete state space (set of all reachable markings) of the ASPN including output symbols attached to state changes can be interpreted as an HnMM. The set of reachable markings and corresponding state changes of the SPN forms the hidden part of the HnMM.

2.1.1 Formal Definition

The formal definition of HnMM uses the state space of a discrete stochastic model as hidden model part. HnMM extend this state space by adding stochastic output to the states or state changes. An HnMM is formally defined as a 5-tuple (S,V,TR,A,Π) with trace O and path Q (Buchholz 2012).

- S is the set of N discrete states of the system.
- V is the set of M output symbols.
- TR is the set of state transitions, where each $TR_i = (dist, b(v), aging)$ can represent multiple state changes, $dist$ describes the continuous probability distribution function that governs a state transition

and $b(v) : V \mapsto [0,1]$ determines the output probability of every symbol for that transition, *aging* is set to true, if the transition has a race age memory policy.

- $A = \{a_{ij}\}_{NxN}$ describes the complete state transition behavior and a_{ij} is the set of transitions that can cause the state transition from S_i to S_j .
- $\Pi = (\pi_1, ...\pi_N)$ is the initial probability vector of the discrete states and π_i is the probability of the system to be in state S_i at time 0.
- O is a sequence of output symbols v_t with time stamps e_t and also called the trace.
- Q is a sequence of system states s_t with time stamps e_t and also called the path.

The model $\lambda = \{A, \Pi\}$ completely describes the dynamic behavior of the system and is therefore often used. For a more detailed formal definition refer to Buchholz (2012).

2.1.2 Proxel-based Evaluation of HnMM

The Evaluation task aims at the computation of the probability of a given output sequence O for a given model λ: $P(O|\lambda)$. Evaluation is often used to determine the most likely model to have generated the trace. If there are several models possible, the one with the highest trace probability is assumed to be the most likely generating model. Krull, Buchholz, and Horton (2011) describe how to compute the Evaluation probability using the so-called Proxel method.

A Proxel $(s, \vec{\tau}, \rho, p)$ is a probability element which contains all relevant information to compute the future development of a system. In case of HnMM Evaluation a Proxel contains the current system state s, the age of all enabled or race age transitions $\vec{\tau}$, an appropriate representation of the path that has lead to the Proxel ρ and the probability of the combination p.

The Proxel algorithm starts with the initial system state(s) and probability 1. In discrete time steps possible follow-up Proxels and state change probabilities are determined based on the current state and age of the active state transitions. (Horton 2002; Lazarova-Molnar 2005) To include the given output sequence, the generated follow-up states are tested for being able to produce the given trace, and are only pursued further if they are valid given the trace. A follow-up state is valid, if there was a symbol, and the current path may emit it, or if there was no symbol and the current path may emit none. All states reached at the end of the given output sequence form the end points of possible system development paths. Adding up their probabilities gives the desired Evaluation probability. (Krull, Buchholz, and Horton 2011)

2.1.3 Proxel-based Decoding of HnMM

The Decoding task is the computation of the most likely path for a given output sequence O and a given model: $argmax_{q_0 \ldots q_U \in \{S_1 \ldots S_N\}^{U+1}} P(Q|O \cap \lambda)$. Krull, Buchholz, and Horton (2011) describe how to compute the Decoding path using the Proxel algorithm. The basic algorithm and modifications are the same as above. Instead of accumulating probability when joining Proxels, the largest probability is taken and the path has to be included in every Proxel. The Proxel with the highest probability at the end of analysis of a given trace is the end point of the most likely generating path. Backtracking from there gives the most likely generating path. A path describes one possible system development over the whole simulation time, it usually contains a list of all state transitions and the corresponding time stamps. This completely describes the system development, knowing the initial state. (Krull, Buchholz, and Horton 2011)

2.2 Reward Modeling

Rewards are used to incorporate continuous measures of interest (e.g., performance, reliability, dependability, performability) in different variants of stochastic Petri nets (Sanders and Meyer 1991; Muppala, Ciardo, and Trivedi 1994). Rate rewards and impulse rewards are the two most common kinds. Rate rewards increase or decrease a real valued variable according to the given system state or Petri Net marking. Examples of rate rewards are the bandwidth of a network link or the storage cost for items in stock. Impulse rewards increase or decrease a real valued variable instantaneously upon the completion of a given activity or firing of a transition. Examples of impulse rewards are repair cost to be paid or profit generated by a sale.

Stochastic Reward Nets (SRN) (Muppala, Ciardo, and Trivedi 1994) use a stochastic Petri net as description of the dynamic process of the system. The reward structure defines which states or transitions influence the reward measure and how they do it. They are used for reliability prediction of systems. Rate rewards depend on the number of tokens in a specific set of places of the net. Impulse rewards are associated with the completion of activities of the SPN and firing of the corresponding transitions.

Sanders and Meyer (1991) augment Stochastic Activity Networks (SAN) with a reward structure. SANs are similar to Petri Nets, but contain probabilistic transition behavior and gates, that can influence or monitor the complete system state. Different types of rewards are considered, but they are all classified as either rate-based or impulse-based. The time behavior of the rewards differs, some are of interest either in system steady state or in the transient phase. We are interested in the transient system behavior and the transient development of the reward measure since we assume to have a trace of samples of a continuous reward measure that was generated by a particular system behavior. Impulse rewards and rate rewards that accumulate over time are applicable, when considering the transient system behavior.

3 HIDDEN NON-MARKOVIAN REWARD MODELS

To extend Hidden non-Markovian Models with rewards, we have to treat the two kinds of rewards independently. Impulse rewards are similarly to discrete symbol outputs in HnMM based on the completion of a state change. However, impulse rewards are not associated with a probability. Rate rewards are fundamentally different from symbol emissions. They are associated to a given discrete state of the system and are accumulated continuously while the system remains in the given state. For HnMRM the rate is always defined based on the model time unit.

3.1 Formal Description of Hidden Non-Markovian Reward Models

Formally an HnMRM is a 6-tuple $HnMRM = (S, TR, A, \Pi, \vec{\rho}, rr)$ with trace O and path Q. S, A, Π and the path Q remain as described in Section 2. The differences to the HnMM definition are the following:

- The HnMRM definition no longer contains a set of symbols. This has been replaced by the vector $\vec{\rho} = (\rho_1, ...\rho_M)$, which is the vector of all continuous reward measures.
- The rate rewards are defined by the function $rr(s, \vec{\rho})$. It describes the changes to the reward vector due to rate rewards in the form of a differential equation $\frac{d\vec{\rho}}{dt}$ depending on the state and current reward values.
- A transition $TR_i = (dist, ir, aging)$ now contains a reward function, instead of the symbol emission probabilities. ir describes the changes to the reward vector $\vec{\rho}$ due to impulse rewards depending on the transition, starting state s, target state s' and current reward values $\vec{\rho}$.
- O is a sequence of measurements of one or more of the reward values $\vec{\rho}$ with time stamps e_t and also called the trace.

These mode changes require changes in the reconstruction algorithms, which will be highlighted in the following section. A complete description is omitted here and will be published in a more extensive format.

3.2 Evaluation and Decoding of HnMRM

Evaluation and Decoding of HnMRM follow the same scheme as in usual HnMM Evaluation and Decoding as described in Section 2. The current reward value is included in the definition of a Proxel (see Section 2.1.2), and updated continuously, as the possible system developments are followed. A path is also included in each Proxel element, which describes the system development that lead to the current Proxel. A path contains all state changes with the time stamps when they occurred, this completely describes the development of the discrete system up to the given point in time and also the development of the continuous measures, since these solely depend on the discrete system state. Essential changes in the Proxel algorithm are the computation of the rate and impulse rewards and the question how to determine valid system development paths.

3.2.1 Computing the Reward Measures

A special reward function computes the changes to the reward values depending on the current reward value and system state or the state change that occurred. Impulse rewards are computed based on the state change that occurred and can also depend on the current system state and reward value. Rate rewards are updated in every time step of the Proxel analysis method and depend on the current system state and the time interval covered by one Proxel time step. Since rate rewards are defined by differential equations, their actual values have to be approximated. We use a simple Eulers integration scheme and an integration time step that is smaller than the target time step (Proxel time step) (e.g. $\frac{\Delta t}{10}$). Eulers method can be used since the Proxel time step is already small and we are using a fraction of this as integration time step.

3.2.2 Determining Valid Paths

In HnMM analysis a path is valid, if the symbol sequence emitted by the computed path matches the trace. Validity is a binary criterion, since the symbols are from a countably finite set (see Section 2).

In HnMRM we are sampling and comparing continuous real-valued rewards, a uncountably infinite set of values. There is no longer a clear boundary between valid and invalid paths. A computed reward will hardly ever exactly match the sample from the trace. One obvious reason are computational errors due to the discretization. The other reason is that computed reward values from the reconstruction and sampled rewards from the trace are only compared at certain discrete points in time, where the Proxel method computes the system state. The reward values can only be compared at the beginning and end of each Proxel time step. If the time stamp of the measurement from the trace lies somewhere in between those points, even an exact reward computation would never hit the correct sample value.

Therefore we define an epsilon value to determine an acceptable difference of the sample from the trace and the computed reward value. If a reward sample falls within a time step, we determine its difference to the computed reward value of the current path at the beginning and end of the time step. If the absolute difference is smaller than epsilon, then the path is considered valid and the Proxel kept. If the difference is larger, the path is invalid and the Proxel discarded. This results in an epsilon neighborhood around the sample value for determining valid Proxels. The path is also kept, if during the time step, the reward value passed the sample value, but is outside of the epsilon environment before and after the time step. A larger epsilon value results in more valid paths, which ensures to retain valid paths in the end. However, a larger value for epsilon also increases time and memory complexity, since more paths are valid, increasing the total number of Proxels that are processed. A good tradeoff for epsilon has to be found, currently on the basis of each model and sequence couple. A general guideline for choosing epsilon will need to be developed in the future. A starting point would be to choose epsilon larger than the maximum possible continuous reward change within one time step of the analysis method.

3.2.3 Cutting Off Irrelevant Paths

In the original Proxel algorithm (Horton 2002; Lazarova-Molnar 2005) the number of Proxels increases exponentially with the length of simulation time. To keep the algorithm feasible, Proxels with a probability below a certain threshold are discarded. This pruning introduces a small error and does not influence the average system behavior. In HnMM analysis we stopped pruning the Proxel tree, since more likely paths might become invalid later on. Keeping all valid paths was often possible, due to the enormous reduction in state space by only following the valid paths (see Section 2). In HnMRM analysis we no longer have a binary criterion for valid paths, and the number of paths is variable; the larger the epsilon-neighborhood, the more paths are generated. Therefore we again prune paths below a given probability threshold. This threshold needs to be defined dynamically, since the absolute probabilities are decreasing in every time step. A small cutoff value ensures that the valid paths do not die out and that we will retain the most likely path in the end, even though during the process it might be less likely than others which become invalid later on. However, a larger cutoff threshold results in increased time and memory complexity. Therefore, a good tradeoff has to be found, currently on the basis of each model and sequence couple.

Thus adapted, behavior reconstruction for HnMRM using Proxels is now possible.

4 EXPERIMENTS

The experiments in this section show that the analysis of HnMRM is possible using Proxels. Examples for sample sequences and output paths are given. The absolute values of path and trace probabilities are converted to a logarithmic scale, since they decrease fast. Both use cases are illustrated by comparing one exemplary path and trace with the path reconstructed by the analysis method. The test cases chosen are representative and the results for other test cases were very similar to the ones described here.

The first example from medicine shows the possible complexity of the continuous model part. It contains two continuous variables related through differential equations, where only one of them can be sampled. The second example from computer gaming furthermore illustrates the behavior of the Proxel method regarding accuracy, time and memory complexity, when varying the sample interval and the epsilon for the detection of valid paths. The experiment varying trace length is not shown here in detail, however the results are commented on in the conclusion.

4.1 Glucose Insulin Metabolism of a Diabetic

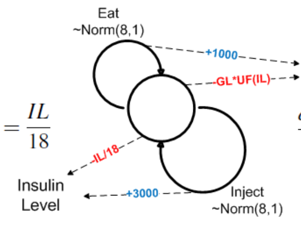
This first example will illustrate the possible complexity of the continuous model part in an HnMRM. We are considering the dynamics of the glucose insulin metabolism of a patient suffering from diabetes type 1, who is no longer capable of producing insulin and has to inject it regularly to retain a bearable blood glucose level. We are using a simplified version of the short term development, meaning the day-by-day development of glucose and insulin level and their interaction. The dynamics of the model have been taken from a model for teaching (Hartlove, Shaffer, and Ragan 2001).

Figure 1 (left) shows the graphical representation of the model. Circles represent states, solid arrows state transitions, dashed arrows rewards. Positive rewards are shown in blue and negative ones in red. There is only one system state and two transitions. Eating happens at randomly distributed intervals of $\sim Norm(8,1)$ hours and increases the glucose level GL immediately by 1000 units (impulse reward). Insulin injection happens at intervals of $\sim Norm(8,1)$ hours and increases the insulin level IL immediately by 3000 units (impulse reward). Rate rewards: the insulin level decreases continuously, as insulin decomposes at a rate depending on the current level. Glucose is “used” depending on its current level and a usage fraction function that depends on the amount of insulin available $UF(IL)$ (see Figure 1 (right)).

The measurement interval of the blood glucose level is $Norm(3,1)$ hours, the insulin level is not monitored and thus the sample sequence only contains glucose level samples.

The experiment was conducted using a sample sequence of length 33 samples which corresponds to 100 hours. The epsilon value was set to 80 and the cutoff threshold was 5. This means that every Proxel with a logarithmic probability below $maximumpathprob - 5$ was discarded, all in all 1123855 Proxels were discarded during the analysis process. These were the best working parameters. A larger cutoff threshold resulted in too many paths and an abort of the analysis due to memory overflow. A larger epsilon value also increase the memory consumption unduly and let to an aborted analysis. A smaller cutoff threshold and epsilon value lead to “dying out” of all valid paths due to no more Proxels being left in the too small epsilon environment. The discretization time step of the Proxel method was chosen to be 1.0 time unit, which was small enough for detailed reconstruction and small enough to not have two symbol emissions within the same time step. Since the time stamps in the reconstructed path can only be multiples of the

$$\frac{dIL}{dt} = \frac{IL}{18}$$



$$\frac{dGL}{dt} = GL * UF(IL)$$

IL	UF(IL)
0	0.0075
1800	0.0105
3600	0.0145
5400	0.019
7200	0.0255
9000	0.033
10800	0.0405
12600	0.049
14400	0.0545
16200	0.0575
18000	0.059

Figure 1: A graphical representation of the insulin glucose metabolism model of a diabetic in the form of a state space with associated rewards (left), including the differential equations describing the continuous behavior of the two reward values and the tabular function describing the usage fraction (right).

discretization time step, all reconstructed time stamps will be whole valued. Computation time of the Proxel analysis was moderate with 15.3 seconds. The example traces were generated using the simulation software AnyLogic6 (A. 2007).

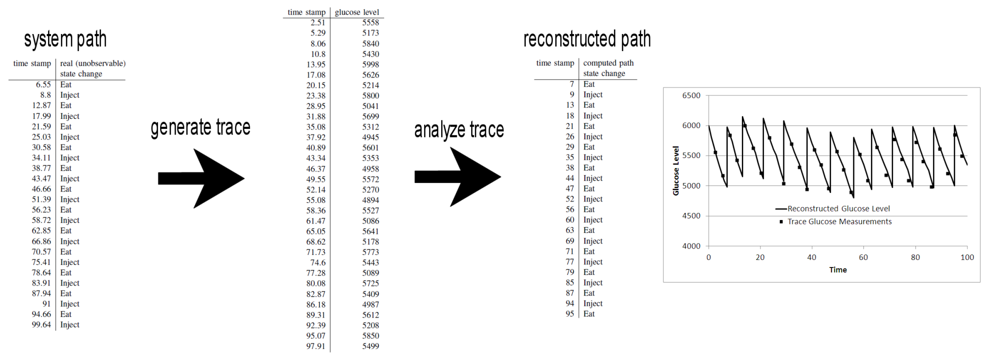
4.1.1 Illustration Experiment

Figure 2 shows an example analysis workflow with paths and trace for the diabetic model. The real system traverses a particular system path during its development (left), this system development is completely described by the state changes that happened and the time stamps when they happened. The real systems insulin and glucose level can be deducted from these state changes. The trace (center) is generated by sampling the blood glucose level of this real system development at random points in time, resulting in a list of glucose level samples with time stamps. The analysis method then reconstructs the path of the system that generated this trace most likely. This reconstructed path (right) is again a list of state changes and time stamps. The chart on the right compares the glucose level development of the reconstructed path (solid line) and the glucose samples from the trace (squares).

The most likely path depicted on the right has a logarithmic probability of -31.10495 which corresponds to a real probability of $3.10E-14$. The conditional probability of the model traversing this most likely path given the observed trace is 0.00062744 while the next most likely paths have conditional probabilities of 0.00062737, 0.00062478, 0.00062475 etc. These likely traces with probabilities which are very close to the most likely path differed slightly in the time stamps of the state changes, but not in their order. The less likely traces showed larger differences in the time stamps.

The deviation of the time stamps of the reconstructed path and the real system paths is small, all except for one are around 1 hour. The existing differences below 1 are due to discretization to the Proxel time step $\Delta t = 1$, which can only result in whole valued time stamps in the reconstructed path. Other differences are due to the stochastic nature of the model and the fact that the actual generating path might not be the most likely one yielding a similar sample sequence. The real path contains one more state change (insulin injection), which is not detectable, because it happens after the last measurement. We can conclude that the reconstructed path is very close to the real generating path for the given sequence.

The experiment was used to illustrate the possible model complexity. Rate rewards are here described as differential equations, where the continuous reward values also depend on each other. Samples are only available for one of the continuous variables, even though both are affected by discrete model behavior, and show an interdependency. Regardless of this, the analysis method is capable of determining the probability of trace for a given model and the most likely path for generating the trace. Even though finding working values for the epsilon environment and the cutoff threshold was not trivial, we could show that the analysis method is capable of dealing with the complexity of the modeling paradigm.



time stamp	real (unobservable) state change
6.55	Eat
8.8	Inject
12.87	Eat
17.99	Inject
21.59	Eat
25.03	Inject
30.58	Eat
34.11	Inject
38.77	Eat
43.47	Inject
46.66	Eat
51.39	Inject
56.23	Eat
58.72	Inject
62.85	Eat
66.86	Inject
70.57	Eat
75.41	Inject
78.64	Eat
83.91	Inject
87.94	Eat
91	Inject
94.66	Eat
99.64	Inject

time stamp	glucose level
2.51	5558
5.29	5173
8.06	5840
10.8	5430
13.95	5998
17.08	5626
20.15	5214
23.38	5800
28.95	5041
31.88	5699
35.08	5312
37.92	4945
40.89	5601
43.34	5353
46.37	4958
49.55	5572
52.14	5270
55.08	4894
58.36	5527
61.47	5086
65.05	5641
68.62	5178
71.73	5773
74.6	5443
77.28	5089
80.08	5725
82.87	5409
86.18	4987
89.31	5612
92.39	5208
95.07	5850
97.91	5499

time stamp	computed path state change
7	Eat
9	Inject
13	Eat
18	Inject
21	Eat
26	Inject
29	Eat
35	Inject
38	Eat
44	Inject
47	Eat
52	Inject
56	Eat
60	Inject
63	Eat
69	Inject
71	Eat
77	Inject
79	Eat
85	Inject
87	Eat
94	Inject
95	Eat

Figure 2: The actual generating path of the real system (left), the trace generated by that path (center) and the most likely path reconstructed by the analysis method for the glucose insulin metabolism model.

4.2 Example from Computer Gaming

As a second example we consider a simple computer game scenario, where a player holds an army and may build up to two bases which can also be destroyed. We only consider the dynamics of one player. The continuous variable is the players army size, while the building and destruction of bases form the discrete process. We are an opponent and have samples of the other players army size only at random intervals through a global statistics. Figure 3 shows a graphical representation of the model. Circles represent states, solid arrows state transitions, dashed arrows rewards. Building a base takes a randomly distributed amount of time $\sim NORM(10,1)$, bases are destroyed by a memoryless process $\sim EXP(0.1)$. Positive rewards are shown in blue, negative ones in red. Rate rewards: when there are no bases, the army size decreases at a constant rate of 5 per time unit, with one base it increases with 5 per time unit and with two bases it increase by 15 per time unit. Impulse rewards: destruction of a base decreases the army size immediately by 50 soldiers. Initially the player has no bases and an army with 100 soldiers.

We defined some standard values for the experiments. Epsilon is set to 10 soldiers, the sample interval is distributed according to $\sim NORM(5,1)$ time units, the length of the test traces is around 200 samples (1000 time units). Test sample sequences of the army size were produced using the simulation software AnyLogic6 (A. 2007). The discretization step of the analysis method was again set to 1.0 resulting in only whole valued time stamps in the reconstructed path. Computation time for the test cases was usually less than one minute (except for very long traces) and can therefore be considered feasible.

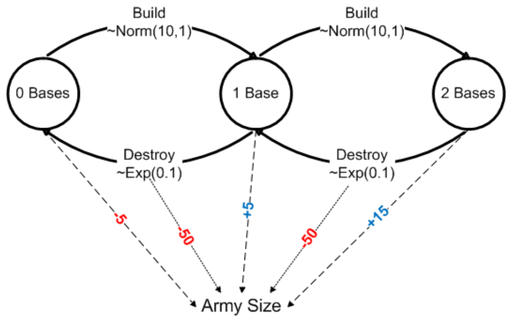
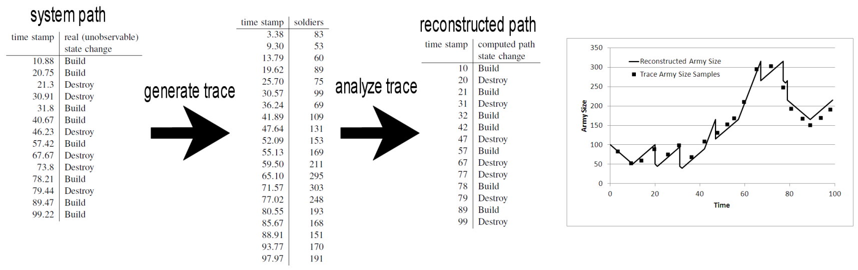


Figure 3: A graphical representation of the computer gaming example model in the form of a state space with associated rewards.

4.2.1 Illustration Experiment

An illustration experiment was conducted using a sample sequence of length 196 (1000 time units). The epsilon value was set to 20 soldiers, meaning that every path that resulted in a reward measure diverging more than 20 soldiers from the sample value of the trace was discarded and not followed further. A smaller value for epsilon lead to all paths dying out, because all valid ones were eventually cutoff. A larger value for epsilon also worked, leading to more valid paths. But it also lead to an unnecessary overhead in computation time, since the rate rewards are constant, and their development moderate and not erratic. Figure 4 shows an example analysis workflow for the computer gaming model. The real system traverses a



time stamp	real (unobservable) state change
10.88	Build
20.75	Build
21.3	Destroy
30.91	Destroy
31.8	Build
40.67	Build
46.23	Destroy
57.42	Build
67.67	Destroy
73.8	Destroy
78.21	Build
79.44	Destroy
89.47	Build
99.22	Build

time stamp	soldiers
3.38	83
9.30	53
13.79	60
19.62	89
25.70	75
30.57	99
36.24	69
41.89	109
47.64	131
52.09	153
55.13	169
59.50	211
65.10	295
71.57	303
77.02	248
80.55	193
85.67	168
88.91	151
93.77	170
97.97	191

time stamp	computed path state change
10	Build
20	Destroy
21	Build
31	Destroy
32	Build
42	Build
47	Destroy
57	Build
67	Destroy
77	Destroy
78	Build
79	Destroy
89	Build
99	Destroy

Figure 4: The actual generating path of the real system (left), the trace generated by that path (center) and the most likely path reconstructed by the analysis method for the computer gaming example model.

particular system path during its development (left). This system development is completely described by the state changes that happened and the time stamps when they happened. Army size and number of bases can be deducted from this list of state changes. The trace (center) is generated by sampling the army size of the real system development at random points in time, resulting in a list of army size samples with time stamps. The Proxel-based analysis method then reconstructs the path that generated the trace most likely. This reconstructed path (right) is again a list of state changes and time stamps. The chart on the right compares the army size development of the reconstructed path (solid line) and the samples from the trace (squares). The most likely path depicted on the right has a logarithmic probability of -262.430326, which corresponds to a real probability of $1.066492e-114$. The conditional probability of the model traversing the most likely path given the observed trace is 0.0802. The next three likely paths have probabilities of 0.0778, 0.0698, 0.0677 etc. These paths differ slightly in the time stamps of the state changes, but not in their order. Less likely traces also differed in the order of the state changes.

This experiment shows that we are able to reconstruct a system development path that could have generated the given sample trace. The order of the state changes in the reconstructed path is very close to the real path. Two state changes that occur in consecutive time steps are switched in their order, and the last state change does not match. These are both not methodological errors, but due to statistical deviations and too few information in the case of the last state change. Furthermore, all except two of the state change time stamps in the reconstructed path are within one time unit of the real time stamp. The existing differences below 1 are due to discretization to the Proxel time step $\Delta t = 1$, which can only result in whole valued time stamps in the reconstructed path. Other differences are due to the stochastic nature of the model and the fact that the actual generating path might not be the most likely one. The small differences between system and reconstructed path show that the reconstruction was successful.

4.2.2 Epsilon Boundary Experiment

This experiment varied the epsilon value, which determines the validity of a path 3.2.2. If the reward value of a reconstructed path is within the epsilon environment of the current trace sample value, the path is valid and followed further. If it is outside, the Proxel is discarded and the path not followed further on. In the experiment epsilon is varied from 20 up to 225. For smaller values the paths “died out”, i.e. all paths generated had reward values outside of the epsilon environment and were discarded.

Figure 5 (left) shows the number of paths still valid at the end of the sample trace with increasing epsilon value. the number of paths increases linearly, since more traces are considered valid in every step. Figure 5 (center) shows the computation time for the analysis with increasing epsilon value. The time complexity also increases linearly, which is to be expected with a linear increase in paths to consider. Figure 5 (right) shows the maximum path probability and cumulative trace probability with increasing epsilon value, both as logarithmic probabilities. Both increase with increasing epsilon value, since more paths considered are valid and their cumulative probability increases. The probability of the most likely path also increases, since the pool of valid paths is larger. Paths considered invalid with smaller epsilon values might have a higher overall probability than the others.

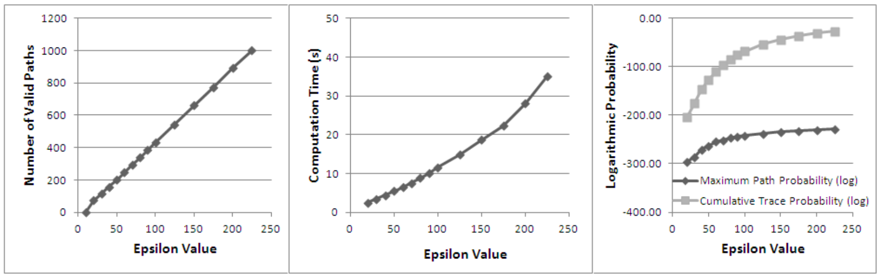


Figure 5: Development of the number of valid paths (left), the overall computation time (center), the maximum path probability and the cumulative trace probability (right) for increasing epsilon value.

The experiment shows that the algorithm behaves as expected when increasing epsilon, and does not show an explosion in time or memory complexity. A general guideline is that epsilon should be chosen large enough for valid paths to “survive” until the end of simulation time, but not much larger to prevent the generation of too many likely paths with larger probability than the real one. This assumes that the actual generating path will be among the first batch of “surviving” paths with small epsilon values.

4.2.3 Sample Interval Experiment

This experiment varied the sample interval used for generating the sample traces between $Norm(5,1)$ and $Norm(30,1)$ in steps of 5 for the mean value. The epsilon value was set to 20 to prevent the “dying out” of paths for larger sample intervals.

Figure 6 (left) shows the the maximum number of concurrent Proxels with increasing sample interval. This corresponds to the maximum number of valid paths in each of the simulation time steps. The number increases linearly with increasing sample interval size, due to the number of possible system developments (Proxels) increasing in every time step of the Proxel method. Only when reading a sample and comparing it to the reconstructed paths, Proxels are discarded. Now there are more Proxel time steps between two successive samples, and the number of concurrent Proxels can grow larger. Figure 6 (center) shows the computation time needed with increasing sample interval. This super-linear relationship is due to the linear increase in concurrent Proxels resulting in an even larger increase in the overall computational effort. Figure 6 (right) shows the maximum path probability and the cumulative trace probability with increasing sample interval. Both increase with increasing interval, since there are more possible paths between two samples, and consequently also paths with larger probability. However, the path with larger probability is not necessarily closer to the actual generating system behavior. The results are therefore less reliable for larger sample intervals, which is to be expected since less data is available.

The experiment shows that more frequent samples enable a more accurate and faster analysis. The accuracy is higher because more frequent samples convey more information about the hidden process. The analysis is faster, because with more frequent samples, the invalid paths can be detected and discarded

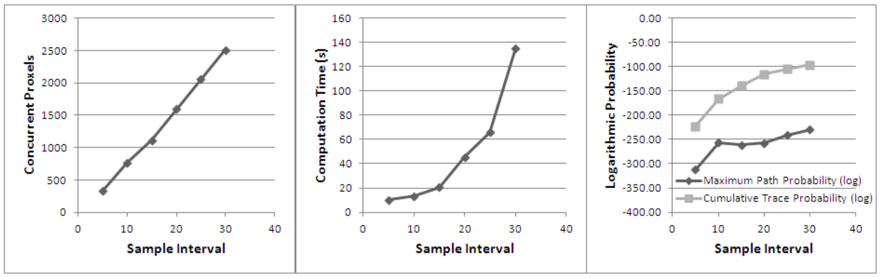


Figure 6: Development of the maximum number of concurrent Proxels (left), the computation time (center), the maximum path probability and the cumulative trace probability (left) for increasing sample interval.

earlier on in the analysis, and and a smaller epsilon value speeds up the analysis further (see Section 4.2.2). Usually we will not be able to influence the size of the sample interval, since we assume to have the samples from the system, but knowledge about useful frequencies are nevertheless useful. Even though the analysis is possible with quite infrequent samples, the results are not reliable, since the most likely path in a larger set of paths generated with less frequent samples will often differ from the actual generating path.

4.2.4 Conclusion of Performance Experiments

The experiments with the computer gaming example model experiment suggest that the Proxel analysis method can find likely generating paths and their probability for a given HnMRM and sample path. Increasing the size of the epsilon environment or sample interval results in an increase in time and memory complexity, but not unduly. An experiment varying the trace length was also conducted but omitted here for reasons of space. When varying the trace length, the algorithm also behaves as expected. It does not show an explosion of path number or time complexity. A linear relationship implies that traces of arbitrary length can be analyzed with acceptable time consumption. Therefore we conclude that the method behaves nicely, and we hope that it also enables the analysis of more realistic model specifications.

One idea for improvement of the tuning process the method is to determine the size of the epsilon environment and the cutoff probability dynamically depending on the number of paths generated. However, the size of epsilon and the cutoff value have an influence on the resulting probability values. Thus when comparing different models, epsilon and the cutoff value should be fixed and the same for both models. Only when trying to determine the most likely generating path, this dynamic tuning can be used.

5 SUMMARY AND OUTLOOK

The paper introduced the new modeling paradigm of Hidden non-Markovian Reward Models, which augment existing HnMM with rewards. This enables the analysis of partially observable hybrid models with hidden discrete behavior, and samples available of some of the continuous variables. HnMRM are formalized and an adaption of HnMM analysis algorithms described. Experiments show that the analysis method is capable of reconstructing hidden system behavior and that no unexpected explosion of computation effort is registered when varying some parameters. An example from medicine illustrates the possible complexity of the continuous part, having ODEs as rate rewards. Still the behavior reconstruction was possible. The extension of virtual stochastic sensors into the continuous realm was successful. We can draw conclusions from samples of continuous variables on unobservable continuous and discrete system behavior.

HnMRM show great potential for the analysis of real systems since these often contain not only discrete parts but exhibit hybrid behavior. To increase algorithm efficiency it will be necessary to find guidelines for certain algorithm parameters such as the size of the epsilon environment, the cutoff value and the sample interval (if we can influence it). A more dynamic tuning of epsilon and the cutoff boundary would speed up decoding. It will also be necessary to test the paradigm using real systems and data. Further ideas for HnMRM are to determine the most needed item of missing data, e.g. answer the question where would we need more samples for a more accurate result. Further extensions of the HnMM paradigm are possible when considering influences of the continuous on the discrete model part. One could also define impulse rewards as samples of continuous or discrete distribution functions.

REFERENCES

Borshchev A. 2007, September. “Multi-Method Simulation Modeling using AnyLogic”. INFORMS Roundtable Fall Meeting, Seattle. www.anylogic.com XJ Technologies.

Bolch, G., S. Greiner, H. de Meer, and K. S. Trivedi. 1998. *Queuing Networks and Markov Chains*. John Wiley & Sons, New York.

Buchholz, R. 2012, March. *Conversive Hidden Non-Markovian Models*. Ph. D. thesis, Otto-von-Guericke-University Magdeburg.

Fink, G. A. 2008. *Markov Models for Pattern Recognition*. Berlin, Heidelberg: Springer.

Hartlove, J., D. Shaffer, and S. Ragan. 2001. “The Maryland Virtual High School CoreModels - Glucose-Insulin Model”. http://mvhs1.mbhs.edu/mvhsproj/glucose/glucosetea.pdf.

Horton, G. 2002. “A New Paradigm for the Numerical Simulation of Stochastic Petri Nets with General Firing Times”. In *Proceedings of the ESS 2002*, edited by R. Rimane, 129–136: SCS European Publishing House.

Ibargüengoytia, P. H., and A. Reyes. 2006. “Constructing Virtual Sensors Using Probabilistic Reasoning”. In *MICAI*, edited by J. G. Carbonell and J. Siekmann, 218–226.

Krull, C. 2008, April. *Discrete-Time Markov Chains: Advanced Applications in Simulation*. Ph. D. thesis, Otto-von-Guericke-University Magdeburg.

Krull, C., R. Buchholz, and G. Horton. 2010, October. “Matching Hidden Non-Markovian Models: Diagnosing Illnesses Based on Recorded Symptoms”. In *Proceedings of ESM 2010. Hasselt, Belgium*, edited by G. K. Janssens, A. Ramaekers, and A. Caris, 133–138.

Krull, C., R. Buchholz, and G. Horton. 2011, February. “Virtual Stochastic Sensors: How to gain Insight into Partially Observable Discrete Stochastic Systems”. In *The 30th IASTED International Conference on Modelling, Identification and Control, Innsbruck. Austria*, edited by R. Fox and M. H. Hamza.

Lazarova-Molnar, S. 2005, November. *The Proxel-Based Method: Formalisation, Analysis and Applications*. Ph. D. thesis, Otto-von-Guericke-University Magdeburg.

Muppala, J., G. Ciardo, and K. S. Trivedi. 1994. “Stochastic Reward Nets for Reliability Prediction”. *Communications in Reliability, Maintainability and Serviceability* 1 (2): 9–20.

Sanders, W. H., and J. F. Meyer. 1991. “A Unified Approach for Specifying Measures of Performance, Dependability, and Performability”. In *Dependable Computing for Critical Applications Vol 4*, edited by A. Avizienis and J. Laprie, 215–237. New York, NY, USA: Springer-Verlag New York, Inc.

AUTHOR BIOGRAPHIES

CLAUDIA KRULL received her German Diploma in Computer Science from the University of Magdeburg, in 2003. She is a research and teaching assistant in the Simulation and Modeling group since then. In April 2008 she defended her PhD Thesis. Her research interests include the analysis of partially observable discrete stochastic systems via virtual stochastic sensors. Her email address is claudia.krull@ovgu.de.

GRAHAM HORTON received his German Diplom in Computer Science from the University of Erlangen in 1989. He received his PhD in Computer Science in 1991 and his Habilitation in 1998 from the same university, in the field of simulation. Since 2001, he is a Professor of Simulation and Modeling at the Computer Science Department of the University of Magdeburg. His research interests include efficient solution algorithms for Markov chains, state space-based simulation methods, innovation and idea engineering. His email address is graham.horton@ovgu.de.