Adapting to Change of Model Transitions in Proxel Based Simulation of CHnMMs

Dávid Bodnár*, Claudia Krull

Institut für Simulation und Graphik, Otto-von-Guericke-Universität Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany; *david.bodnar.ovgu@gmail.com

SNE 33(1), 2023, 9-16, DOI: 10.11128/sne.33.tn.10632
Selected ASIM SST 2022 Postconf. Publication: 2023-02-01;
Received Revised Improved: 2023-02-27; Accepted: 2023-03-10
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Virtual Stochastic Sensors (VSSs) [1] aim to provide insight into stochastic processes by producing statistically relevant estimates of non-measurable system properties. During behavior reconstruction of these discrete stochastic systems the internal system state changes are often described as time-homogeneous distribution functions, as in Conversive Hidden non-Markovian Models (CHnMMs). However, the system behavior might change over time or the sample, used for the model creation, might not describe the system accurately. In [2] and [3] we have shown that detecting these changes is possible, yet the resource consumption for the re-estimation of the model was a clear problem. In this paper we present a solution to that problem by replacing the used statistical tests with Kernel Density Estimation (KDE) and by integrating the hidden model description into the proxel-based state space simulation method. By using the Change Adaptation Algorithm (CAA) this paper shows that adapting to runtime changes is possible, while preserving parameters on transitions where no change occurs. The algorithm was tested with 5 different types of Probability Density Functions (PDFs) which showed accurate results. By using the CAA one is able to construct adaptive models for behavior reconstruction without the need to fully parametrize the model. In this way, loss of modeling accuracy in the model construction process can be significantly decreased.

Introduction

VSSs were introduced in [4]. They are tools to reconstruct the behavior of partially observable processes in discrete stochastic systems. Constructing VSSs relies in practice heavily on manually provided knowledge about the system. But what happens if that information becomes outdated or inaccurate during the model construction or over time? What happens if there are flaws in the model construction of the VSS?

To overcome this limitation this paper introduces the CAA, which describes how a proxel-based analysis of CHnMMs can be extended to tune the stochastic parameters of the system model during runtime. The implementation utilizes KDE to re-estimate the state change distributions in every time step based on historical runtime information. In this way potentially different system models are available in a given time step to provide the most accurate model and trace estimation at the end of the behavior reconstruction.

The paper is a proof of concept to analyze whether change adaptation in this way is possible to provide a more realistic coupling between simulation models and the real world and whether such adaptation can be maintained during the VSS's lifetime.

1 Related Work

Measuring information in complex systems has often physical or financial limitations, which might be resolved using Virtual Sensors (VSs) [5]. For stochastic systems, by combining VSs with stochastic processes, a so-called VSS can be constructed to measure statistically relevant estimates of non-measurable system parameters. One of these possible stochastic processes is called CHnMM [6], which can be analyzed by the proxel-based analysis method.

In this section, a brief overview will be given of the previous work on VSSs and KDE with which we are extending the concept to make change adaptation possible. Additionally, the energy distance will be introduced in a few words, hence this was used during the model evaluation.

Conversive Hidden non-Markovian Model The Hidden Markov Model (HMM) [7] is a well researched technique to analyze directly not observable models through probabilistic symbol emissions. The HMMs assume memoryless model state changes, which is why the concept of Hidden non-Markovian Model (HnMM) was introduced in [8] to extend the HMMs with state changes governed by arbitrary continuous distribution functions. Using HnMMs one is able to define time dependence between different system states.

[6] introduced CHnMMs as a subclass of HnMMs where all state changes of the hidden process of interest emit a symbol for the observer, making additional performance optimizations possible. In this paper, the implementation of the CAA was restricted to CHnMMs to make the proof of concept analysis easier to interpret and to reduce the number of interference factors.

Both CHnMMs and HnMMs try to solve, similarly to the HMMs, the so-called evaluation (finding the probability that a given trace was generated by the model) and decoding tasks (finding the most likely generator state sequence). However, instead of using the Baum-Welch algorithm [7] or the Viterbi algorithm [7] the so-called proxel-based analysis is used which is briefly introduced in the next paragraph.

Proxels-Based Analysis To reconstruct the state space of stochastic processes it is possible to use the so-called proxel-based analysis [9] introduced in [10]. Using this technique one is able to construct container like „probability elements" (proxels) which store all relevant information of a defined discrete simulation state. These are, as shown in Equation 1, the current system state (m), the transition age vector (τ), the probability of the current state (p) and the current timestamp (t). But of course, it can be further extended, for example, with the generator path.

$$P_x = (m, \tau, p, t) \tag{1}$$

The analysis uses discrete timesteps to track the state changes during the process. The connection between a parent proxel and its children in the next time step is characterized by the Hazard Rate Function (HRF) in Equation 2 which describes the rate of probability that a specific state change will happen in the next timestep if it has not happened yet.

$$H(\tau) = \frac{f(\tau)}{1 - F(\tau)} \tag{2}$$

The network of parent and child proxels in the simulation domain construct the so-called proxel tree, which tracks all possible system states in discrete time steps. To prevent state-space-explosion and to provide acceptable simulation times, impossible or very unlikely proxels are pruned.

Kernel Density Estimation KDE was introduced in [11] and [12]. The idea behind the KDE is to place small kernels $K()$ on samples X_i in the domain and use their aggregated sum as a PDF, as written here:

$$\hat{f}(x,b) = \frac{1}{nb} \sum_{i=1}^{n} K\left(\frac{x - X_i}{b}\right) \tag{3}$$

Where n is the number of elements in the sample, and b is the so-called bandwidth parameter, which is a free smoothing parameter to „stretch" the kernels to a possibly optimal PDF. The bandwidth selection is a well researched topic on its own. [13] gives a detailed overview about the different techniques and their limitations.

There are multiple techniques to improve the accuracy of the estimated PDF. One of them is using variable KDE, discussed in [13] and [14], where every single kernel gets its own bandwidth through the weights w_i. In this way, one is able to create „spikes" in the PDF where the underlying samples are more dense and still preserve smooth tails and junctions, which is challenging with real world data and a constant bandwidth.

$$\hat{f}(x,b) = \frac{1}{nb} \sum_{i=1}^{n} \frac{1}{w_i} K\left(\frac{x - X_i}{b w_i}\right) \tag{4}$$

As the CHnMMs are working in the time domain, boundary correction [15] can be used to restrict the KDE computation to positive numbers.

KDE has the advantage over regular PDF definitions that one is not bounded to a defined class of PDFs, because a wide variety can be approximated with KDE as long as a suitable number of samples (n) is available and one is able to choose an appropriate bandwidth (b).

Energy distance The energy distance [16] (Equation 5) describes a statistical distance based on New-

ton's potential energy between two independent random samples (X and Y) described by the Cumulative Distribution Functions (CDFs) F and G. The resulting distance is 0 if and only if $F = G$.

$$D^2(F,G) = 2\mathbb{E}||X-Y|| - \mathbb{E}||X-X'|| - \mathbb{E}||Y-Y'|| \quad (5)$$

Compared to the popular statistical tests like the Wald-Wolfowitz Runs Test, the Kolmogorov-Smirnov Test [17], etc. [18] the energy distance is superior in quantifying differences [19] and it is also closely related to the Cramér von Mises distance [20], which is often used to quantify the difference between distribution functions.

In order to interpret the energy distance, it needs to be normalized as written in [21] (Equation 6). The expression D_n is bounded between zero and one and it is equal to zero if and only if the samples X and Y have the same distribution.

$$D_n = \frac{D^2(F,G)}{2\mathbb{E}||X-Y||} \quad (6)$$

This metric can be used to quantify deviation between two distribution functions so in this paper it was used for describing the deviation of the model transition from the ground truth in the evaluation section.

2 Change Adaptation Algorithm

The CAA utilizes the KDE ideas described in Section 1 to construct the transition distributions in every time step. This extension has the additional positive effect that the state transitions are not directly linked to any elementary distribution function and its prelimitary properties.

For the KDE computation a given amount of historic samples are used, which are stored for every transition separately using sliding windows. The basic idea is to couple the proxels using these samples with the model definition, in a way that every single proxel encapsulates its own model representation of the CHnMM. As a kernel the standard normal distribution was selected, which has an infinite support, resulting in theory in KDEs accepting new samples from the whole domain, even outside of the current PDF with a very small but non-zero probability.

By using KDE, the CAA extends the content of a single time step. Before, the time step t_i contained n probable system states from which the most probable one(s) will survive. With CAA the number of proxels in a given time step represent $m \leq n$ different models with n different system states from which the most probable one(s) will survive.

Upon a model drift on a specific transition, the new samples will be automatically assigned to the history of the most probable transition as a result of Equation 2. By doing so, the new samples push out the old model samples from the history with every new time step and this automatically results in a reconfiguration of the CHnMM model through the KDE. As a result, the model is able to adjust itself to model changes with a given delay defined by the length of the history vector and the mean of the new samples.

Implementation To implement the CAA the transition age vector τ gets replaced by a transition vector $\mathbf{T}$ which holds the age τ and the history vector $\mathbf{h}_n$ for every single transition in the CHnMM. The history vector holds the last n firing time samples for the transition T_i.

$$P_x = (m, \mathbf{T}(\tau, \mathbf{h}_n), p, t) \quad (7)$$

The basic mechanisms of the CAA are shown on a flowchart in Figure 1. The values of the history vectors are used as a basis for the KDE. As a bandwidth selector a univariate direct plug-in selector, described in [13], was used.

In a given timestamp t_i, a KDE is computed for every active transition of every proxel. The state change probabilities are computed based on these KDEs and the algorithm generates all child proxels based on the possible state changes for the time step t_{i+1}. The history vector of all the fired transitions gets altered based on the current timestamp and age criteria. At the end, very unprobable child proxels are pruned from the proxel tree before the next time step begins.

The CAA needs an initial model injected upon start. To do that, every transition of the first proxel at t_0 is preseeded with some kind of history. We used as preseed n quantile samples with a n^{-1} step regular grid from an assumed CDF. The samples were reordered according to the PDF so that the most probable sample gets removed first during the sliding window approach.

KDE is a resource intensive computation. We used a look-up-table to speed up the algorithm with a residual error in the PDF of 10^{-5}. This error does not cause a significant difference in the end results unlike other computations in the background, like the Kingsbury-Rayner formula [7], that have a much higher computa-

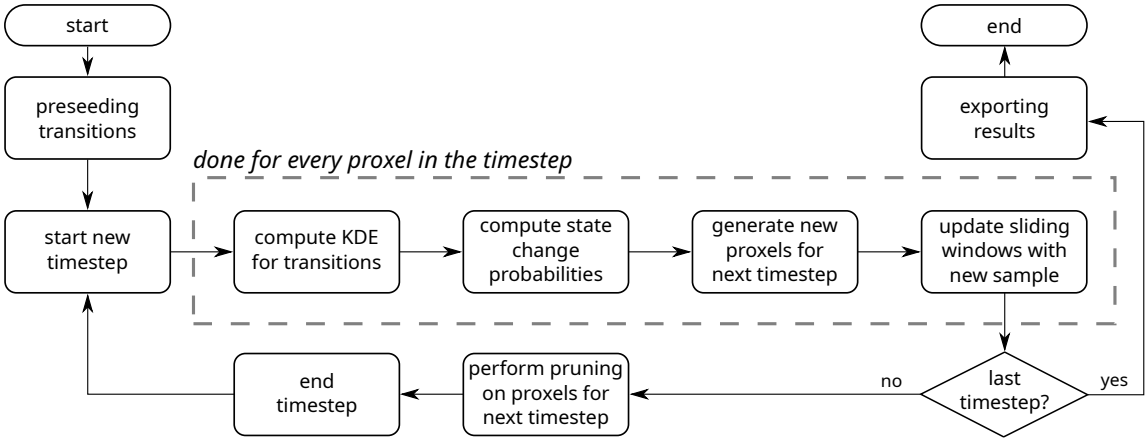

Figure 1: Flowchart of the Change Adaptation Algorithm

tional error. This discretization has the negative drawback that very strong changes in the model result in 0 probability instead of a very small one which can result in a died out proxel tree. This means, that in practice the algorithm will need a roughly usable preseed at the beginning.

Parameters There are two major parameters impacting the results. One of them is the size of the history vector represented by the window size. The other one is the amount of improbable proxels getting removed from the proxel tree in every time step controller by the pruning.

Window size The window size, so the length of the history vector, defines the memory of a given transition. By choosing higher values the algorithm gets more robust against outliers and KDE becomes more accurate. However, this also results in more computational complexity and the adaptation to a new change gets slower as new samples need more time to push out the old ones from the window. Choosing a too small window size can result in a faster computation, but the model will be affected by any interference on the input side of the simulation.

Pruning There are two major strategies for pruning away unlikely states. One of them is keeping only a given number of most-likely proxels, which might result in more inaccurate models. The other one is defining a so-called pruning threshold which will delete proxels if their probability gets below a given value. In this paper we restricted ourselves to the second approach.

$$p(P_{pruned,t_i}) < r \max(p(P_{x,t_i})) \tag{8}$$

The pruning threshold r, as it can be seen in the Equation 8, is defined by a fixed probability ratio of the most probable proxel in the time step and the proxels which are considered too unlikely. Choosing a high value results in a proxel tree without any real diversity in a single time step. In most of these cases a single model becomes prevalent and if that becomes impossible, the tree dies out. However, if one chooses a too low value then the state space explodes and the algorithm will run too slow (as long as enough RAM is available).

$$r = \begin{cases} r_{min} & \text{if } \#P_{x,t_i} < \#P_{min} \\ r(\#P_{x,t_i}) & \text{if } \#P_{min} \leq \#P_{x,t_i} \leq \#P_{max} \\ r_{max} & \text{if } \#P_{x,t_i} > \#P_{max} \end{cases} \tag{9}$$

The concept of variable pruning threshold was introduced, as fixed pruning thresholds did not fulfill all the requirements of the CAA. The basic idea is (Equation 9) to define a mathematical function between a fixed minimum and maximum pruning threshold based on the number of the proxels in the given time step and use that to define a dynamic transition between those values. This makes it possible to significantly reduce the execution time and to prevent state space explosion while maintaining accurate results and model estimation.

3 Experiments

Experiment setup To validate the CAA, a basic academic example was used, originally introduced in [6]. In this model two production lines get merged before a common quality tester. Both lines produce faulty products with a given probability.

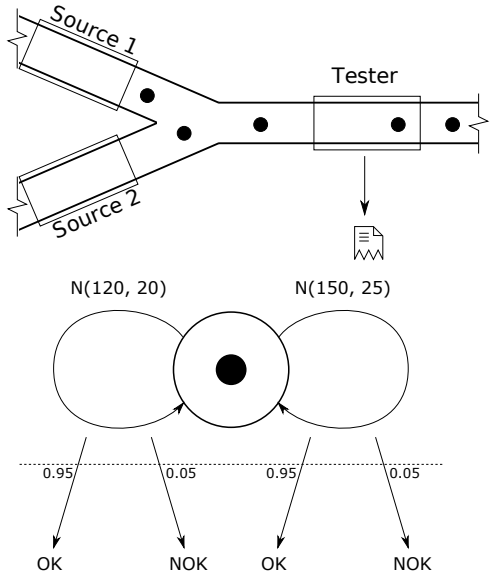

Figure 2: Quality tester example [6] and its ASPN [1]

The system can be described with the ASPN [1] in Figure 2. The symbol emission probabilities were chosen to be equal on both transitions in order to eliminate possible information gain through the asymmetric probabilities.

To validate the CAA, different PDFs were assigned to the transitions describing the production line sources. In a single experiment, Source 1 and Source 2 had always the same distribution type as ground truth to simplify the comparison of the results and to be able to neglect errors introduced by the KDE, as the error appears in both distribution estimations in a similar way. Figure 2 shows the original model parametrization, which was used to preseed the algorithm upon execution start. If not normal distribution was used, the distribution of the specific type under analysis had the same location and scale parameters as mentioned in the figure.

The input data for the experiment was generated in a way that Source 2 received ground truth information, while for Source 1 samples from a changed distribution were generated. For easier comparison of the results, Scipy's [22] location-scale(-shape) parametization was used. Please, be aware that this parametization differs from the standard academic notation. Five distribution function types were analyzed: normal, uniform, exponential, lognormal and Weibull distribution. The change consisted of location parameters on a grid of $[60, 150]$ with a step size of 10, while the scale parameter was tested between $[5, 30]$ with a step size of 5. For PDF types lognormal and Weibull an additional shape parameter was used to test a wider range of possible changes. For every parameter combinations, 20 random experiments were executed.

Parameter selection To analyze the impact of the variable parameters on the experiment results, the parameters window size and pruning threshold were analyzed in a way that the parameter under test was changed on a regular grid while the other parameter was held constant. The results can be seen in Figures 3 and 4.

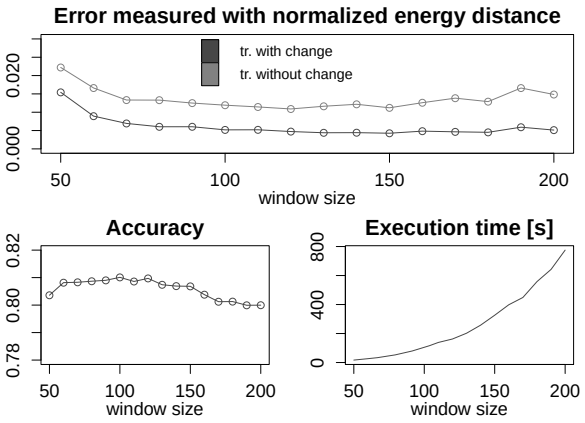

Figure 3: Impact of window size. The error is measured between the ground truth and the model distributions reached at t_{end}. The accuracy plot shows the rate of correctly classified symbols.

In Figure 3 one can see that by increasing window size the execution time increases drastically, as the KDE computation has $O(n^2)$ [13] computational complexity. However, we can also see that choosing a greater window size does not automatically lead to better performance as described in Section 2. Similarly, if we choose the window size too small then the outliers have a negative impact on the results. The optimal window size lies in this case between $120 - 130$ so we chose 125 for the experiments.

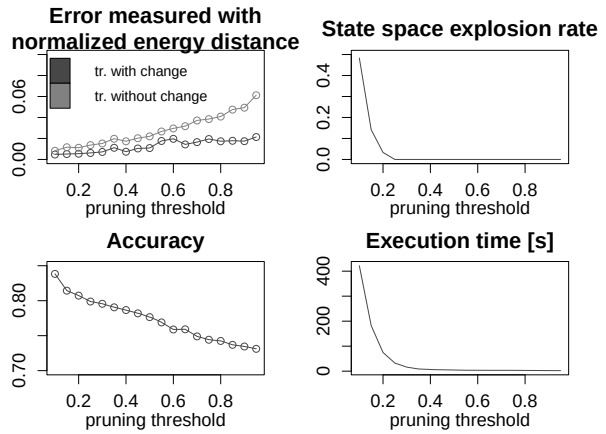

Figure 4: Impact of pruning threshold. The error is measured between the ground truth and the model distributions reached at t_{end}. The plot with the state space explosion shows the rate of failed experiment executions due to out of RAM. The accuracy plot shows the rate of correctly classified symbols.

In Figure 4 one can see that with increasing pruning threshold the model and the symbol classification become more inaccurate. This is a direct impact of reduced diversity in a time step. The optimal pruning threshold can be found around the value 0.2, however, there is a significant risk of experiencing state space explosion.

$$r(\#P_{x,t_i}) = 0.15(\log(\#P_{x,t_i}) - 3)^2 + 0.1 \qquad (10)$$

To overcome these limitations variable pruning was introduced between $n = [10^3, 10^5]$ and $r = [0.1, 0.7]$ with Equation 10. This made it possible to reach the same accuracy results as with the static threshold 0.2 and prevent state space explosions accurately on the cost of $\approx 40\,\%$ increase in mean execution time.

Experiment results Table 1 shows how well the symbols were classified. The precision and recall values were also included as there is a significant difference between the number of symbol emissions on the different transitions. Generally, it can be said, that the classification performs quite well, however, the transition with change has a higher true positive rate ($\geq 80\,\%$ compared to $\geq 75\,\%$). This is due to the higher emission rate on the transition 1.

Surprisingly, even in case of the exponential distribution, we get accurate results. Without the CAA, as written in [2] and [3], concurrent exponential transitions were always a problem for the proxel-based simulations as the HRF is in this case constant and all symbol emissions get assigned to the transition with the higher HRF. KDE cannot fully reproduce the exponential distribution in the background, in this way, by violating the theory of the exponential distribution we get practically usable results.

Table 2 shows the model errors at the end of the experiment. There is an expected amount of faulty results in the experiment which was basically predefined by the setup. The defined parameters in Subsection 3 already show that there are cases when the two transitions will (with a very high probability) merge. These expectations were documented in the „Tr. merge" column of the table with ranges in square brackets. One can see that these expectations were always fulfilled, but the values were in the lower range. This means that the CAA was able to distinguish in some cases between distributions which were very close to each other, however, this is most probably pure luck.

Of course, when an algorithm tries to adjust itself to a changing environment, the classification might flip, which in our case means that the transitions switch places. We label a case as a transition flip if the mean of the resulting transitions flip compared to the ground truth. This happened only in less then 5% of the cases, except the lognormal distribution. It means, that the CAA is able to accurately adjust itself to new models while it is able to maintain a constant distributions.

If we take a look at on the bad, or better said not fully accurate, models, the same statement can be made. A model on a transition was considered to be bad, if the normalized energy distance (Equation 6) between the KDE estimate and its target distribution exceeded the value 0.03. Please, note that this is not a p-value. A flipped model should automatically result in a bad model in this table, so that is responsible for a significant amount of the bad models. However, please bear in mind that there are model definitions very close to each other so merged models with very slight differences in the ground truth might not result in a bad model with or without a transition flip. This happened in the case of the normal and the exponential distributions. Generally, it can be said that in $\geq 91\,\%$ at least one, and in $\geq 84\,\%$ both transition models were accurate.

Dist. type	Accuracy	Precision 1	Precision 2	Recall 1	Recall 2
Normal	$80.60\% \pm 0.43\%$	$83.52\% \pm 0.41\%$	$75.50\% \pm 0.50\%$	$83.17\% \pm 0.43\%$	$75.95\% \pm 0.48\%$
Uniform	$89.92\% \pm 0.45\%$	$91.71\% \pm 0.36\%$	$86.59\% \pm 0.66\%$	$91.25\% \pm 0.38\%$	$86.69\% \pm 0.66\%$
Exponential	$84.15\% \pm 0.37\%$	$86.67\% \pm 0.37\%$	$80.05\% \pm 0.39\%$	$85.82\% \pm 0.38\%$	$81.25\% \pm 0.39\%$
Lognormal	$80.86\% \pm 0.28\%$	$80.18\% \pm 0.33\%$	$81.05\% \pm 0.25\%$	$83.82\% \pm 0.28\%$	$77.36\% \pm 0.29\%$
Weibull	$88.07\% \pm 0.24\%$	$89.27\% \pm 0.24\%$	$85.96\% \pm 0.27\%$	$89.30\% \pm 0.23\%$	$85.93\% \pm 0.27\%$

Table 1: Model state classification results. The values with 1 in the header, refer to the transition where a model change was introduced, while the values with 2 in the header refer to a transition where the model was kept constant.

Dist. type	Bad model tr. 1	Bad model tr. 2	Bad model both	Tr. flips	Tr. merge
Normal	1.44%	2.03%	1.11%	1.81%	4.54%, [2.78% – 5.56%]
Uniform	3.29%	3.29%	2.92%	0.74%	1.39%, [1.39% – 2.78%]
Exponential	5.28%	3.52%	1.81%	2.96%	3.94%, [2.78% – 5.56%]
Lognormal	15.15%	12.45%	8.03%	7.39%	5.31%, [2.92% – 5.84%]
Weibull	7.39%	5.82%	4.81%	3.98%	3.41%, [2.38% – 4.76%]

Table 2: Model estimation results. The values with 1 in the header, refer to the transition where a model change was introduced, while the values with 2 in the header refer to a transition where the model was kept constant. The values in [ ] in the Transition merge column refer to the expected rate of transition merges due to the experiment setup.

One can also see that the lognormal distribution results are by far the worst for the CAA. However, theoretically there should be no significant limitation for this distribution type. A high amount of bad models can be traced back to transition flips. After a deeper analysis, it turned out that 90% of the models classified to be bad can be traced back to heavily tailed lognormal distributions. This is most probably a result of inaccurate KDEs and a too small window size. We assume that by increasing the window size these model problems would disappear.

The CAA is highly parallelized in a time step and it has a low memory footprint (around 1-2 GB-s) compared to the previous algorithm described in [2] and [3]. As a result, the average execution time varied between 1-5 minutes which is superior to the old algorithm. However, this is still around $10-20x$ higher than the same VSS without the CAA.

4 Conclusion

The experiments show that the CAA is efficiently able to track and to adapt to model changes in CHnMMs positively affecting the symbol classification accuracy. Doing so does not affect the capability of withstanding changes on transitions where no change occurred.

By using the algorithm, the construction of CHnMM based VSSs is possible without deep analysis of the model parameters. It is enough to inject a distribution independent sample upon start.

The CAA also resolves some old practical issues with Proxel-based simulations, like the problem of concurrent exponential distributions, limitation through fixed distribution types and runtime changes of the model, which were not manageable before.

There is no theoretical limitation that would speak against the application of the CAA. It results in a very high probability of better evaluation and decoding results than manually parametrized models. Therefore, its usage is generally encouraged. However, if higher execution times are not acceptable, one might want to compromise and enable the functionality only on selected transitions.

Applications and Future Research The CAA opens up more accurate simulation possibilities for non-stationary models, like production lines and other human-influenced system.

Future research possibilities include the generalization of the CAA for VSS related problems beside CHnMMs. The transition flip problem could be reduced by adding penalty terms for transition movements. Additionally, the variable pruning can be evaluated as general pruning algorithm for VSSs.

Acknowledgement

The authors thank [22], [23] and [24] for making their work available on a free and open-source basis.

References

[1] Krull C. *Virtual Stochastic Sensors: Formal Background and Example Applications: Reconstructing the Behavior of Partially Observable Discrete and Hybrid Stochastic Systems*. Shaker. 2021.

[2] Bodnár D. *Change Detection of Model Transitions in Proxel Based Simulation of CHnMMs*. Magdeburg: Otto-von-Guericke-Universität. 2016.

[3] Bodnár D, Krull C, Horton G. Change Detection of Model Transitions in Proxel Based Simulation of CHnMMs. In: *Analytical and Stochastic Modelling Techniques and Applications*, edited by Thomas N, Forshaw M. Cham: Springer International Publishing. 2017; pp. 32–46.

[4] Buchholz R, Krull C, Horton G. Virtual Stochastic Sensors: How to gain insight into partially observable discrete stochastic systems. *The 30th IASTED International Conference on Modelling*. 2011;.

[5] Wilson E. Virtual sensor technology for process optimization. 1997. Presentation at the ISSCAC 1997.

[6] Buchholz R. *Conversive Hidden non-Markovian Models*. Magdeburg: Otto-von-Guericke-Universität. 2012.

[7] Fink GA. Markov Models for Pattern Recognition. In: *Advances in Computer Vision and Pattern Recognition*. 2014; .

[8] Krull C, Horton G. Hidden non-Markovian Models: Formalization and solution approaches. *6th Vienna International Conference on Mathematical Modelling*. 2009;.

[9] Lazarova-Molnar S. *The Proxel-Based Method: Formalisation, Analysis and Applications*. Magdeburg: Otto-von-Guericke-Universität. 2005.

[10] Horton G. A new paradigm for the numerical simulation of stochastic Petri nets with general firing times. In: *Proc. European Simulation Symp. ESS'02*. 2002; .

[11] Rosenblatt M. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*. 1956;27(3):832 – 837.

[12] Parzen E. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*. 1962;33(3):1065–1076.

[13] Gramacki A. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. Cham: Springer International Publishing. 2018.

[14] Salgado-Ugarte I, Perez-Hernandez M. Exploring the Use of Variable Bandwidth Kernel Density Estimators. *Stata Journal*. 2003;3:133–147.

[15] Jones M. Simple boundary correction for kernel density estimation. *Statistics and Computing*. 1993;3:135–146.

[16] Rizzo ML, Székely GJ. Energy distance. *WIREs Computational Statistics*. 2016;8(1):27–38.

[17] Magel RC, Wibowo SH. Comparing the Powers of the Wald-Wolfowitz and Kolmogorov-Smirnov Tests. *Biometrical Journal*. 1997;39(6):665–675.

[18] Arnold TB, Emerson JW. Nonparametric goodness-of-fit tests for discrete null distributions. *R Journal*. 2011;3(2).

[19] Rizzo ML. A test of homogeneity for two multivariate populations. *Proceedings of the American Statistical Association, Physical and Engineering Sciences Section*. 2002;.

[20] Szekely G. E-statistics: The Energy of Statistical Samples. 2002.

[21] Szekely G, Rizzo M. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*. 2013;8.

[22] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat , Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 10 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020; 17:261–272.

[23] Rizzo M, Szekely G. *energy: E-Statistics: Multivariate Inference via the Energy of Data*. 2022. R package version 1.7-9.
URL `https://CRAN.R-project.org/package=energy`

[24] Carreño CR. vnmabus/dcor: Version 0.5. 2020.
URL `https://doi.org/10.5281/zenodo.3996697`