MARKOV CHAINS AGGREGATION USING DISCRETE EVENT OPTIMIZATION VIA SIMULATION

Laurent Capocchi
Jean-Francois Santucci

SPE UMR CNRS 6134
University of Corsica
Campus Grimaldi
20250 Corte, France
{capocchi, santucci}@univ-corse.fr

Bernard P. Zeigler

RTSync Corp
12500 Park Potomac
Potomac, MD
zeigler@rtsync.com

ABSTRACT

Markov chains are an important form of stochastic system representation. Recent modeling techniques supporting discrete-event Markov model composition make it easy to build large Markov chains that are difficult to analyze due to state space explosion. Lumpability is a well known condition that allows reduction in state space but its strict requirements inhibit potential use. In this paper, we introduce a discrete-event based framework to construct and aggregate Markov chains using a relaxed form of lumpability (quasi-lumpability) with an associated metric. Based on state partitions we describe a search methodology to select an optimum partition according to a metric that allows comparing Markov chains based on their respective steady states. Such optima are computed using a discrete-event optimization via simulation approach. The framework enables us to enhance our understanding of the space of finite Markov chains and the search complexity of the space.

Keywords: DEVS, Markov chains, lumpability, Optimization via Simulation.

1 INTRODUCTION

Markov chains (Kemeny and Snell 1976) are one of the simplest forms of stochastic dynamic system that allows to model stochastic dependencies. The main interest of a Markov chain, beyond the ability to simulate a stochastic system, is to be able to predict its behavior. The Markov chain can be viewed as a device for calculating a probability distribution over the states of the chain. When the right conditions are met (Markov chain that is aperiodic and irreducible), the simulation of a Markov chain converges towards a limit stationary distribution (called steady state) which allows the prediction.

However in order to deal with real world complex systems applications the Markov chains have to be reduced. Recent modeling techniques supporting composability (Zeigler and Nutaro 2016; Zeigler, Nutaro, and Seo 2017) make it easy to build large Markov chains that are difficult to digitally analyze due to the state space explosion occasioned by the cross product of component states. The analysis consists essentially to compute the stationary distribution. It is therefore required to develop techniques to reduce the size of the matrices (and hence the complexity) before analyzing the associated Markov chain while keeping a guarantee on the quality of results. In order to reduce a Markov chain, a set of approaches have been defined based on reduction of the state set of the Markov chain involving properties such as of exact lumpability and quasi-lumpability (Kemeny and Snell 1976). Exact lumpability is a property of a Markov chain associated with a specific partition of the state space into groups (or macro-states). This property is equivalent to the exact aggregation: it ensures that the process (chain) obtained after aggregation according to the partition remains a Markov chain. In practice, few chains have this property and it is helpful to use a lower threshold: quasi-lumpability.

The proposed approach uses the quasi-lumpability technique that generalizes the exact lumpability but does not guarantee accurate results. Several methods have been proposed to calculate bounds on the stationary distribution of a quasi-lumpable chain. Once we are able to generate from a Markov chain, reduced Markov chains using lumpability or quasi-lumpabality technique based on partition of states, we have to select the optimum partition according a metric allowing to compare Markov chains based on their respective resulting steady states. The optimum is computed using an Optimization via Simulation (OvS) approach involving the DEVS (Discrete EVent system Specification) formalism (Zeigler, Muzy, and Kofman 2018). The framework also enables us to enhance our understanding of the space of finite Markov chains and the search complexity of the space. The proposed approach has been implemented using the MS4Me (Zeigler and Sarjoughian 2013) and DEVSimPy (Capocchi et al. 2011) DEVS modeling and simulation environment. A pedagogical example is used in order to detail the proposed approach while the validation step is based a real case example involving a complex neural network application.

The rest of the paper is organized as follows: Section 2 deals with the background and related work of the proposed approach involving Markov chains and their main properties, the DEVS formalism and its FPDEVS extension. The DEVSimPy and MS4Me frameworks are also introduced. In Section 3 the DEVS manipulations of Markov chains involving simulations and reduction based on exact and quasi-lumpability. Two approaches have been defined and implemented in order to deal with exact lumpability and quasi-lumpability using the DEVS formalism in DEVSimPy and MS4Me frameworks. Both of them are illustrated on concrete examples. Section 4 concerns the DEVS optimization via simulation approach used to obtain the optimum reduction of a given Markov chain while Section 5 is focused on the exploration of the lumpability space of large Markov chain and opens further research developments. Section 6 gives the conclusions and future work.

2 PRELIMINARIES AND RELATED WORK

Section 2.1 introduces the Markov chain theory with the definitions involved in the Lumpability process, the optimization via simulation process and the presentation of the FD-DEVS formalism with two associated framework: MS4Me and DEVSimPy. Section 2.2 discusses related work.

2.1 Preliminaries

2.1.1 Markov Chain Theory and Lumpability

In the Markov theory, a Markov chain is a stochastic process $X(t)$ defined in a finite state space $S = \{1,2,\ldots,n\}$ (Kemeny and Snell 1976). It generates a series of observations, X. The Markov property is that the probability distribution over the next observation depends only upon the current observation. Let $p_{i,j}$ the probability that the next observation is j ($X(t+1) = j$) given that the current observation is i ($X(t) = i$). Basically, these transition probabilities is represented through a transition matrix P. In this paper, we only considerer ergodic chains (the transition matrix must be irreducible and acyclic) $M = (S,P,\pi)$. An ergodic Markov chain as a unique stationary distribution π (called steady state) such that $\pi = \pi P$. However when dealing with large Markov chains, the manipulation is difficult because of the large number of states. Lumpability is used in order to work at a high level of probabilistic hierarchy using a partition of the set of states. Let's introduce two definitions to define the quasi-lumpability and the lumpability based on probability matrix.

Definition 1. *Let P be a stochastic matrix associated with an ergodic Markov chain. The quasi-lumpability on the partition (m macro-states) $C_1, C_2, \ldots, C_m$: For all macro-states C_i,C_j,*

$$\max_{l_1,l_2 \in C_i} | \sum_{m \in C_j} P_{l_1,m} - \sum_{m \in C_j} P_{l_2,m}| = E(i,j) \leq \varepsilon$$

if $\varepsilon = 0$ we have the classic definition of the exact lumpability.

Definition 2. *If M is a Markov chain; S the set of states, π is the steady state and P the transition matrix. The Markov chain (S,P,π) is lumpable using a partition $L = \{C_1, C_2, \ldots, C_m\}$ on S if there exists a matrix $Q = (L, \widehat{P}, \widehat{\pi})$ of order m, such that for all $i, j \in \{1, \ldots, m\}$ and for all $k \geq 0$ it holds,*

$$\widehat{P^k}(i,j) = \frac{\sum_{i' \in C_i}[\pi(i') \sum_{j' \in C_j} p^k(i',j')]}{\widehat{\pi}(i)} \tag{1}$$

where $\widehat{\pi}(i) = \sum_{i' \in C_i} \pi(i')$.

The lumpability is used to perform model reduction. Moreover, a metric is needed to compare two Markov chains with different state space: N for the original Markov chain and M for the lumped Markov chain obtained after the partitioning process (with $dim(M) < dim(N)$). This process is based on a partition function ϕ that gives the relation between N and M. In this paper we choose to use the K-L divergence rate that consider only chains with same dimension defined as follow. We introduce also the definition of the partition function.

Definition 3. *For two stationary Markov chains (N, π, P) and, (N, π', P') defined on the same state space N the Kullback–Leibler divergence (K-L divergence) rate is a measure of the difference between these two Markov chains and is given by the following formula inspired from (Rached, Alajaji, and Campbell 2004):*

$$R(P||P') = \sum_{i,j \in N} \pi_i P_{i,j} log(\frac{P_{i,j}}{P'_{i,j}}). \tag{2}$$

Definition 4. *Let $N = \{1,2,...,n\}$ and $M = \{1,2,\ldots,m\}$ with $m \leq n$. A partition function $\phi : N \rightarrow M$ is a surjective function from N onto M. Fork $\in M, \phi^{-1}(k)$ denotes the k^{th} group in N.*

In order to compare a Markov chain P (with a state space N) and a lumped Markov chain Q (with a state space M obtained a partition function ϕ), we need to lift the Markov chain Q to the original state space N. The lifted Markov chain is denoted $\widehat{Q}$ and is defined as follow.

Definition 5. *Let ϕ the partition function on N; let M denote the range of ϕ and Q denote a lumped Markov transition Matrix on M obtained from P defined on N; let π denote the stationary distribution of P. Then π-lifting of Q is defined as:*

$$\widehat{Q}_{i,j}(\phi) = \frac{\pi_j}{\sum_{k \in \psi(j)} \pi_k} Q_{\phi(i)\phi(j)} \qquad i,j \in N \tag{3}$$

where $\psi(j) = \phi^{-1} \circ \phi(j)$ denotes the set of states belonging to the same group as the j^{th} state in N.

We illustrate the manipulation of Markov chains on a weather forecasting example. Figure 1 depicts the probabilistic finite state automata involving the three states Sunny, Rainy and Cloudy. When the weather is Sunny (rst_n in Figure 1 goes to the initial state), the probability to have the next day Rainy is 0.1, Cloudy is 0.9 and Sunny is 0.0. The same applies to the other two states.

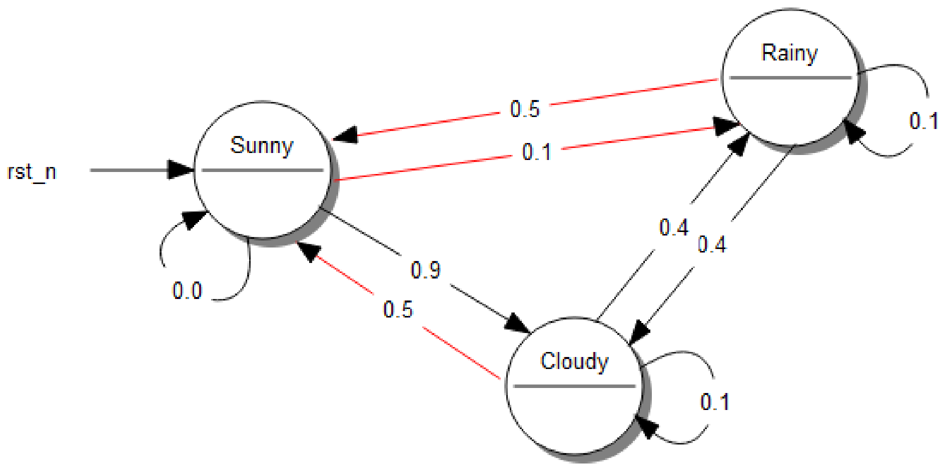

Figure 1: Probabilistic finite state automata of the weather forecasting example.

The Markov chain is based on the following transition matrix:

$$P = \begin{bmatrix} 0.0 & 0.1 & 0.9 \\ 0.5 & 0.1 & 0.4 \\ 0.5 & 0.4 & 0.1 \end{bmatrix}.$$

The steady state π can be computed by solving the equation $\pi P = \pi$. We have $[\pi_0, \pi_1, \pi_2] = [0.33330, 0.23070, 0.4359]$ and the probability that today is cloudy is 0.4359. In order to illustrate the lumping process, we choose a partition $L = \{\{Sunny\}\{Rainy, Cloudy\}\}$ and the corresponding lumped Markov chain $Q = (L, \widehat{P}, \widehat{\pi})$ obtained from Def. (2):

$$\widehat{P} = \begin{bmatrix} 0.0 & 1.0 \\ 0.5 & 0.5 \end{bmatrix} \qquad and \qquad \widehat{\pi} = [0.3333, 0.6666].$$

2.1.2 Optimization via Simulation

Optimization via simulation is a structured approach to determine optimal settings for input parameters (i.e., the best system design), where optimality is measured by a function of output variables associated to a simulation model (Swisher et al. 2000). One of the main features of simulation is that one can change the parameters of a simulation model easily and try to observe the system performance under different sets of parameters. Therefore, it is natural to try and find the set of parameters that optimizes the system. This set of parameters are determined using an optimization model after evaluating an objective function. Figure 2 depicts the coordination between the optimization model and the simulation model.

An optimization problem concerns the minimization or maximization of an objective function $f(x)$ where x is a vector $x = \{x_1, ..., x_n\}$ of n decision variables. A solution of the optimization problem is an assignment of

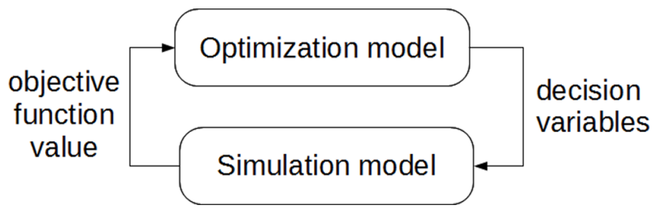

Figure 2: Coordination between the optimization and simulation models.

specific values to x. All solutions form a set X of solutions $(x \in X)$. When decision variables are continuous, $x \in \Re^n$. There are a number of important subclasses of optimization problems depending on the ability to define constraints on the decision variables. When there is no constraints on the decision variables, the unconstrained maximization formulation can be used as follow:

$$\max_{x \in \Re^n} f(x) \tag{4}$$

with the objective function $f : \Re^n \rightarrow \Re$. the decision variables are used to construct object criteria as a mathematical function.

2.1.3 FP-DEVS Formalism

The DEVS formalism was introduced in the seventies (Zeigler 1976) for modeling discrete-event systems in a hierarchical and modular way. DEVS formalizes what a model is, what it must contain, and what it doesn't contain (experimentation and simulation control parameters are not contained in the model. Moreover, DEVS is universal and unique for discrete-event system models. Any system that accepts events as inputs over time and generates events as outputs over time is equivalent to a DEVS model. DEVS allows automatic simulation on multiple different execution platforms, including those on desktops (for development) and those on high-performance platforms (such as multi-core processors). With DEVS, a model of a large system can be decomposed into smaller component models with couplings between them. DEVS formalism defines two kinds of models: (i) atomic models that represent the basic models providing specifications for the dynamics of a sub-system using function transitions, (ii) coupled models that describe how to couple several component models (which can be atomic or coupled models) together to form a new model. Finite Probabilistic DEVS (FP-DEVS) (Zeigler, Nutaro, and Seo 2016; Seo et al. 2015) is an extension of Finite Deterministic DEVS (FD-DEVS). FP-DEVS allows the transition out of a state to be one of a finite set of possible states where the choice is made probabilistically whereas FD-DEVS makes an internal transition from a state to a state. FP-DEVS extends the FD-DEVS specification of the internal transition (Seo et al. 2015).

The Figure 3 illustrates an example of FPDEVS which has one input (?) and two outputs (!). 'B' state can be changed to two states according to probability values at the internal transition stage.

MS4Me software (Zeigler and Sarjoughian 2013) implements DEVS M&S with a Java computer language on the Eclipse environment. In the MS4Me software, the atomic model can be constructed from a state diagram designer which is a graphic user interface to help users visualize behaviors of the atomic model with symbols. Also, it can be created using a restricted DEVS Natural Language (DNL) whose keywords are highlighted in a DNL editor in the MS4Me software. The atomic model component contains states, transitions, and ports (input and output). The coupled model is an instance of System Entity Structure (SES) (Zeigler, Seo, Coop, and Kim 2013, Zeigler, Seo, and Kim 2013) that describes a system with entities relationships and coupling information in restricted natural languages. Finite state Markov chain model

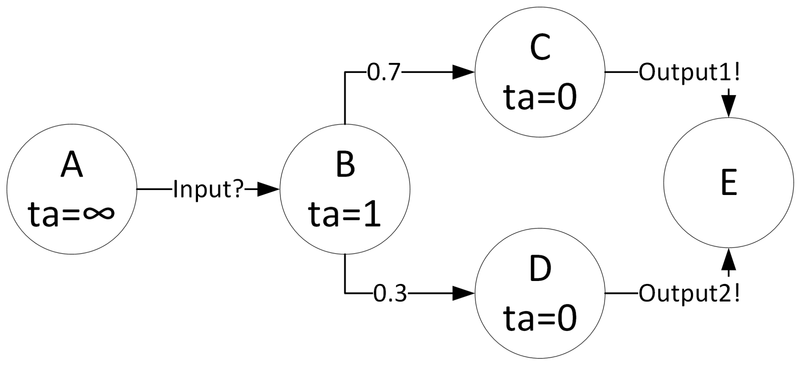

Figure 3: Example of FPDEVS automata.

classes (Zeigler et al. 2013; Zeigler, Seo, and Kim 2013) , with both discrete and continuous time bases, have been implemented in MS4Me using the above described FP-DEVS capabilities. In this paper, we are interested by the CTM (Continuous Time Markov) implementation (Zeigler and Nutaro 2016). The CTM interpretation employs the probabilities associated with internal transitions as specifications of their time advances. Thus the smaller a probability of transition is, the longer is the associated time to its next occurrence (on average) and the less likely it is to be selected as the transition to take. The DEVSimPy (Capocchi et al. 2011) framework is an open source software (under GPL V.3 license) dedicated to the DEVS modeling and simulation of complex systems. It has been implemented in the Python language with the wxPython graphic library, Scipy, Numpy and no other dependencies. The aim is to provide a Graphical User Interface (GUI) for the M&S of PyDEVS models (Van Tendeloo and Vangheluwe 2014). PyDEVS is an Application Programming Interface (API) allowing the implementation of the DEVS formalism in Python language.

2.2 Related Work

Aggregation based model reduction of Markov chains is a widely investigated topic and the use of lumping techniques to reach this goal is not new in the field of Markov systems modeling. Finite Markov chains and lumpability term have been introduced by Kemeny and Snell at the beginning of 60's (Kemeny and Snell 1960) and a fundamental limitation to the reduction of Markov chains via aggregation has been well mathematically resumed in (Kotsalis and Shamma 2012).

Recent papers have been published dealing with higher-order lumpability based on information-theoretic cost functions for finding a partition and approximating the process on this partition by a k-th order Markov chain (Geiger and Wu 2017). Arruda, Fragoso, and Ourique (2017) introduce a new decomposition algorithm for calculating the steady state probability of finite state space ergotic Markov chains. An interesting work combining decomposition and reduction techniques has been published in order to avoid computation of lumping equivalences on intractably large Markov chains (Müllner, Theel, and Fränzle 2013). Xu, Beck, and Salapaka (2014) and Xu, Salapaka, and Beck (2014) propose a method based on deterministic annealing approach for aggregating large Markov chains, where they are viewed as weighted directed graphs.

As can be seen from the related works, the lumpability process can be based on the K-L measure (Geiger and Wu 2017; Deng, Mehta, and Meyn 2011; Geiger et al. 2015) but there are not many works that combine traditional lumpability algorithm based on K-L divergence rate and the Optimization via Simulation process to improve the lumpability process by selecting the optimum partition.

3 DEVS-BASED MARKOV CHAINS LUMPABILITY

This Section concerns with DEVS based Markov chains lumpability and shows how to deal with Markov chains using the DEVS formalism in both the DEVSimPy and MS4Me frameworks.

The DEVS formalism offers the following benefits when dealing with Markov chains:

- Event driven simulation can be performed in order to compute the steady state. Two approaches will be proposed: (i) an analytic approach using DEVSimPy; (ii) a simulation based approach using MS4Me.
- The DEVS formalism involves the use of experimental frames, which would permit to integrate the loops required by OvS to find the optimum Markov chains reduction into a generic DEVS specification.
- The DEVS formalism fits very well with the treatment of Semi Markov chains (Levy 1954) which require the involvement of temporal problems (Zeigler et al. 2018). We plan to extend the approach here to Semi Markov models in future work.

Finite state Markov chain model classes (Pinsky 1972; Kemeny and Snell 1976) , with both discrete and continuous time bases, have been implemented in MS4Me using the above described FP-DEVS capabilities. MS4Me provides a class MarkovMat and a transformation from the FP-DEVS CTM to its MarkovMat corresponding matrix representation. A MarkovMat instance is in fact a discrete time FD-DEVS with a state vector as a state variable and an internal transition function which multiplies the vector by the matrix to update it at every time step. A test for reaching an equilibrium (steady) state is included in the internal transition. Markov Chains have been implemented in DEVSimPy using the pykov package (https://github.com/riccardoscalco/Pykov) that offers a set of functions allowing to deal with Markov Chain manipulation (class Chain with methods like steady(), stochastic(), move(), etc.). A pykov-based "Markov" DEVSimPy library has been developed in order to propose the Markov chain theory in DEVSimPy environment. Moreover, this library has been extended with a set of atomic models dedicated to implement the functionalities defined in Formula (1), (2) and (3) and presented in Section 2.2. Using the OvS approach already presented in (Santucci and Capocchi 2015) and the new Markov library proposed in this paper, we are able to find the optimum partition for Markov chain reduction with DEVSimPy.

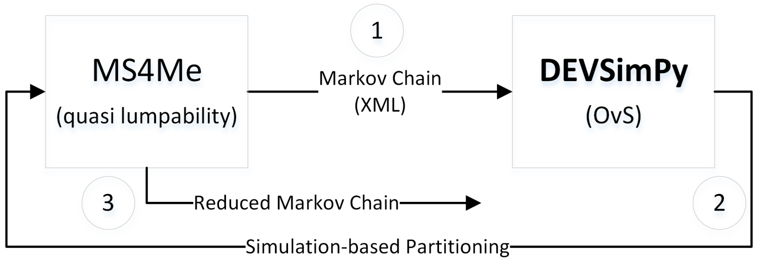

Figure 4: Partition-based Markov chain reduction using MS4Me and DEVSimPy.

MS4Me is a powerful software to model Markov chain with a simulation-based approach. It permits to perform Markov chain reduction using the quasi-lumpability (Def. (1)) from a given state partitioning. Since DEVSimPy allow to generate the optimum partition, we propose to combine MS4Me and DEVSimPy through Markov chains that are shared with XML files (Figure 4). The partition-based Markov chain reduction involves three steps: (i) the Markov Chain matrix is built using the MS4Me environment and transmitted to the DEVSimPy framework through an XML file; (ii) the Simulation-based Markov reduction described in detail in section 4 is applied within DEVSimPy and the resulting partitioning is transmitted to the MS4Me framework using an XML file; (iii) Finally using MS4Me, the corresponding reduced Markov chain is computed and the new steady state as well as the steady state error (between the initial Markov chain and the reduced one) are computed.

4 SIMULATION-BASED MARKOV MODEL REDUCTION

Let (N,P,π) be a given stationary Markov chain. The m-partition problem, is to find the partition function $\phi : N \rightarrow M$ and the optimal aggregated Markov chain (M,Q,ω) such that $\Re^{(\phi)}(P||\widehat{Q})$ (defined in Def. (3) with $\widehat{Q}$ defined in Def. (5)) is minimized:

$$\min_{\phi} R^{(\phi)}(P||\widehat{Q}). \tag{5}$$

In (Deng et al. 2009), due to the large number of partition functions, the m-partition problem is addressed by combining a parameterization of randomized partition policy and a simulation based gradient descent algorithm instead of using Rank and Selection (R&S) or other classic optimization approaches (Swisher et al. 2000). We consider a class of policies described in terms of a parameter vector θ. Simulation is employed to estimate the gradient of the performance metric with respect to θ, and the policy is improved by updating θ in a gradient direction. At the end the θ vector allows to deduce which states have to be aggregated (those having a θ_i similar). In the parameterization approach, a vector $\theta = [\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N] \in \Re^N$ is introduced (Marbach and Tsitsiklis 1998) and associated with a randomized partition policy $\nu_\phi(i,\theta)$ where $\sum_\phi \nu_\phi(i,\theta) = 1$. The randomized partition policy $\nu_\phi(i,\theta)$ has been defined as follow:

$$\nu_\phi(i,\theta) = \frac{1}{1+e^{M\theta_i}} I_{\phi(i)=1} + \sum_{j\in[2,m]} \frac{(\frac{j-1}{m-1})e^{M\theta_i}}{1+e^{M\theta_i}} I_{\phi(i)=j} \tag{6}$$

where $I_{\phi(i)=j}$ is 1 if $\phi(i)=1$ and 0 otherwise. The parameter $M>0$ is a positive constant defined by the user (Deng et al. 2009). In this paper, the simulations have been executed with $M=0.0005$. The parameter $\nu_\phi(i,\theta)$ is combined with the simulation-based gradient descent algorithm which is used to obtain θ and then analyze it to deduce the optimum partition. The effectiveness of the gradient descend algorithm in this case where the decision variable is continuous has been proved in (Deng et al. 2009). The update of $\theta^{(t)}$ is performed at each iteration of the simulation of the Markov chain (N,P,π) that generates a single simple path $X(t)_{t=0}^{T}$ where T is the number of iterations and $X(t) \in N$.

$$\theta^{(t+1)} = \theta^{(t)} - \frac{1}{t} \nabla g_{X(t)}(\theta^{(t)}) \tag{7}$$

where $\nabla g_{X(t)}(\theta)$ and the one step cost $g_i(\phi)$ are computed as follow:

$$\begin{cases} \nabla g_i(\theta) = \sum_{\phi\in\Phi} \nabla \nu_\phi(i, \theta g_i(\phi) & \forall i \in N \\ g_i(\phi) = \sum_{i\in N} P_{i,j} log\left(\frac{P_{i,j}}{\widehat{Q}_{i,j}(\phi)}\right) & \text{(derived from Def. (3))} \end{cases} \tag{8}$$

At the end of the simulation-based gradient descent algorithm, the optimal partition is given by merging all similar values θ_i. The result of the proposed algorithm is illustrated in the previous example of the weather prediction introduced in the Section 2.2.

We give the results of the simulation-based gradient algorithm obtained on the pedagogical example of Section 2.2. Concerning the weather Markov model (Figure 1), the required parameters to performed the simulation based gradient descendent are as follow:

- the positive constant include in the Formula 6: $M = 0.0005$
- the number of iterations: $T = 100$
- the initial state: $X(0) =$ 'Sunny'
- the initial parameter vector: $\theta^{(0)} = [[1.0, 1.0, 1.0)]$
- the set of partitions: L = [('Cloudy'), ('Rainy','Sunny'), ('Rainy'), ('Cloudy','Sunny'), ('Sunny'), ('Cloudy','Rainy')]

At the end of the simulation, $\theta^{(100)} = [1.0010, 0.9999, 0.9988]$, and the probabilities of states being in the first group are $\nu_{\phi=[1,1,1]}(.\theta^{(100)}) = [0.49, 0.50, 0.50]$. We can conclude that the optimal partition function is $\phi* = [1,2,2]$ and the optimal partition is similar to the one presented in Section 2.1.2.

5 LARGE SPACE MARKOV CHAIN CASE STUDY

The case study concerns a CTM neuron network model. The network involves a set of interconnected blocks of neurons. Figure 5 described the DEVS coupled model of CTM Neurons executed and aggregated using quasi-lumpabiblity. Starting from blocks of N neurons which are all to all connected with probabilistic indegree. We have to point out that each block's activity is measured using the following scale: 0, low, medium, high. The interconnected blocks of Neurons are modelized by net of Matrix components. CTM Markov are derived from global state transitions. The reduction of the Markov chains (lumping process) is performed by state partitioning. From the associated CTM matrix models, prediction of steady state probabilities are obtained.

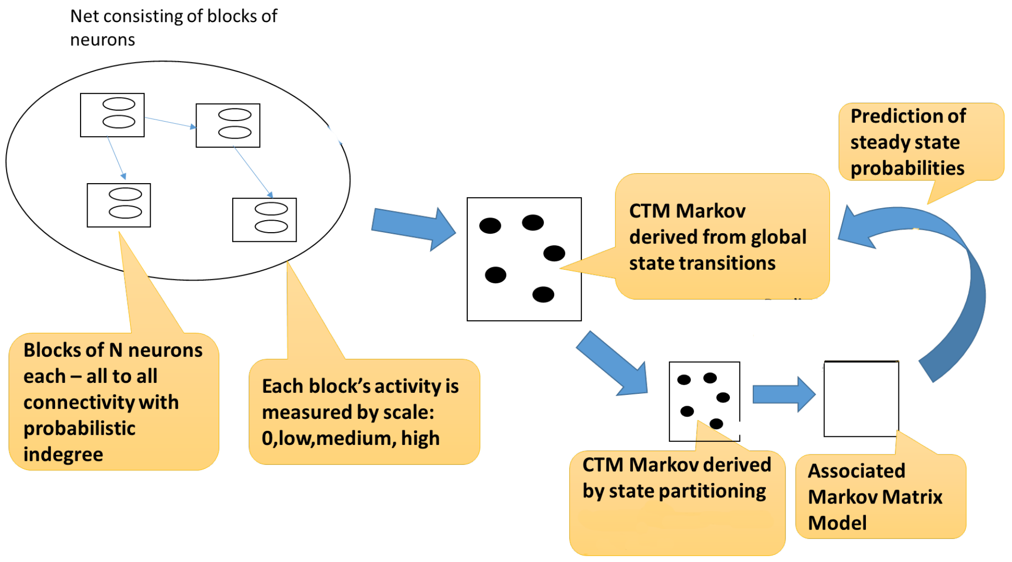
Net consisting of blocks of neurons
Prediction of steady state probabilities
CTM Markov derived from global state transitions
Blocks of N neurons each – all to all connectivity with probabilistic indegree
Each block's activity is measured by scale: 0,low,medium, high
CTM Markov derived by state partitioning
Associated Markov Matrix Model

Figure 5: Coupled Model of CTM Neurons executed and aggregated into Markov Model using quasi-lumpability.

The lumping process proceeds in 2 stages: first, the block activities are lumped into levels and the global state of the net is abstracted into vectors of the form $a_1, a_2, \ldots, a_i, \ldots, a_n$ is the number of blocks and a_i is the activity level of block i. Second the vectors are reduced to vectors of the form $(l_1, l_2, \ldots, l_i, \ldots, l_m)$ where m is the number of levels (here 3) and l_i is the number of blocks having activity level i.

The algorithm developed for Markov chains reduction (by finding the optimum partitioning using a simulation-based gradient descent approach) has been performed and the results are described bellow.

Table 1: Results of Random/Simulation-based Lumping of CTM Neuron Nets.

	Random			Simulation-based		
Model #	Base / Lumped Size	Lumped Std	Std Error	Base / Lumped Size	Lumped Std	Std Error
1	13/13	0.0	0.0	13/12	0.079	0.010
2	24/20	0.031	0.10	24/15	0.051	0.037
3	48/30	0.053	0.38	48/12	0.076	0.081

The lumpability and error metrics just described were applied to the lumping aggregations described in Section 4 for various configurations of the CTM Neuron Nets described there. The results, described in Table 1, show three cases of two different lumping where the overall values of the metrics are shown in the last two columns (Lumped Std and Std Error). Std stands for steady state and it is important to consider this measure in order to compare the two lumping presented approaches. The first lumping corresponds to a random partitioning of the initial base model performed using MS4Me while the second lumping is obtained using the simulation-based partitioning performed using DEVSimPy. The first case was for low level of connectivity which resulted in a small (13 states) chain which could be reduced, only on the second reduction (Simulation-based reduction), by one state. Connectivity was increased to obtain the next 2 cases resulting in larger chains with non-trivial reductions. This 48 states chain was reduced using the random partitioning to 30 and using Simulation-based partitioning to 12 states with the first reduction resulting in relatively high error (0.38) and the second reduction in relatively low error (0.081). Furthermore we observed that cyclic behavior of the net was preserved by the lumping. Such preservation is a more qualitative property than steady state error and one open to study more deeply in further research.

6 CONCLUSION AND FUTURE WORK

In this paper we propose a simulation based algorithm for reducing large Markov chains implementation using the DEVS formalism. The proposed reduction is based on the lumpability of Markov chain (both exact and quasi-lumpability). We have implemented the lumpability process in the MS4Me framework. Furthermore we presented an approach in order to select the optimum partition based on the following concepts: (i) the use of the K-L metric to compare Markov chains; (ii) the development of a parametrization of randomized partition policy; (iii) the development in the DEVSimPy framework of a simulation-based gradient descent algorithm involving the K-L metric as one step cost and allowing to compute a parameter associated with the previously introduced randomized partition policy that points out the optimum partition. A case study has been presented allowing to enhance the understanding of the lumpability space.

Future works will deal with three directions: (i) we will seek to extend the proposed approach to reduce a given Markov chain by trying to guide the simulation towards relevant areas of the state space when looking for the optimum partition (it is not the case in the proposed paper) - we plan to use a complementary approach based on learning (Neural Nets or Support Vector Machines) in order to minimize the risk of not exploring relevant parts of the state space - (ii) we plan to consider Semi Markov models whose behavior depends on the time value (the probability of a state change depends on the amount of time that has elapsed since entry into the current state) since the DEVS formalism's properties fit very well with the resolution of temporal Markov problems - (iii) we plan to further explore the properties of the lumpability space, extending to larger chain sizes and considering other properties such as preservation of transition entropy and cyclic/absorbing steady state behavior as well as implications for search method complexity and more extensive statistical validation of the results. This task imply the use of the Parallel PDEVS algorithm in order to perform the aggregation for a large scale Markov model.

REFERENCES

Arruda, E. F., M. D. Fragoso, and F. O. Ourique. 2017, Dec. "Multi-Partition Time Aggregation for Markov Chains". In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 4922–4927.

Capocchi, L., J. F. Santucci, B. Poggi, and C. Nicolai. 2011. "DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems.". In *WETICE*, pp. 170–175, IEEE Computer Society.

Deng, K., P. G. Mehta, and S. P. Meyn. 2009, Dec. "A Simulation-Based Method for Aggregating Markov Chains". In *Proc. of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 4710–4716.

Deng, K., P. G. Mehta, and S. P. Meyn. 2011, Dec. "Optimal Kullback-Leibler Aggregation via Spectral Theory of Markov Chains". *IEEE Transactions on Automatic Control* vol. 56 (12), pp. 2793–2808.

Geiger, B. C., T. Petrov, G. Kubin, and H. Koeppl. 2015, April. "Optimal Kullback-Leibler Aggregation via Information Bottleneck". *IEEE Transactions on Automatic Control* vol. 60 (4), pp. 1010–1022.

Geiger, B. C., and Y. Wu. 2017, Feb. "Higher-Order Kullback-Leibler Aggregation of Markov Chains". In *SCC 2017; 11th International ITG Conference on Systems, Communications and Coding*, pp. 1–6.

Kemeny, J., and J. Snell. 1960. *Finite markov chains*. Springer.

Kemeny, J. G., and J. L. Snell. 1976. *Finite Markov Chains*. Springer.

Kotsalis, G., and J. S. Shamma. 2012, Oct. "A fundamental limitation to the reduction of Markov chains via aggregation". In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1449–1454.

Levy, P. 1954. "Processus semi-markoviens". In *Proceedings of the International Congress of Mathematicians*, Volume 3, pp. 416–426.

Marbach, P., and J. N. Tsitsiklis. 1998. "Simulation-based optimization of Markov reward processes". In *Proc. of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, Volume 3, pp. 2698–2703 vol.3.

Müllner, N., O. Theel, and M. Fränzle. 2013. "Combining decomposition and reduction for state space analysis of a self-stabilizing system". *Journal of Computer and System Sciences* vol. 79 (7), pp. 1113 – 1125.

Pinsky, M. A. 1972. "An Introduction to Probability Theory and Its Applications, Vol. 2 (William Feller)". *SIAM Review* vol. 14 (4), pp. 662–663.

Rached, Z., F. Alajaji, and L. L. Campbell. 2004, May. "The Kullback-Leibler divergence rate between Markov sources". *IEEE Transactions on Information Theory* vol. 50 (5), pp. 917–921.

Santucci, J. F., and L. Capocchi. 2015, January. "Optimization via Simulation of Catchment Basin Management Using a Discrete-event Approach". *Simulation* vol. 91 (1), pp. 43–58.

Seo, C., B. P. Zeigler, D. Kim, and K. Duncan. 2015. "Integrating Web-based Simulation on IT Systems with Finite Probabilistic DEVS". In *Proc. of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, DEVS '15, pp. 173–180. San Diego, CA, USA, Society for Computer Simulation International.

Swisher, J., P. Hyden, S. Jacobson, and L. Schruben. 2000. “A survey of simulation optimization techniques and procedures”. In *Proc. of the Simulation Conference, Winter*, Volume 1, pp. 119 –128 vol.1.

Van Tendeloo, Y., and H. Vangheluwe. 2014. “The Modular Architecture of the Python(P)DEVS Simulation Kernel (WIP)”. In *Proc. of the Symposium on Theory of Modeling & Simulation - DEVS Integrative*, DEVS ’14, pp. 14:1–14:6. San Diego, CA, USA, Society for Computer Simulation International.

Xu, Y., C. L. Beck, and S. M. Salapaka. 2014, Dec. “Aggregation of Markov chains: an analysis of deterministic annealing based methods”. In *53rd IEEE Conference on Decision and Control*, pp. 6591–6596.

Xu, Y., S. M. Salapaka, and C. L. Beck. 2014, Oct. “Aggregation of Graph Models and Markov Chains by Deterministic Annealing”. *IEEE Transactions on Automatic Control* vol. 59 (10), pp. 2807–2812.

Zeigler, B., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation. 3rd Edition.* Orlando, FL, USA, Academic Press, Inc.

Zeigler, B. P. 1976. *Theory of Modeling and Simulation*. Academic Press.

Zeigler, B. P., and J. J. Nutaro. 2016. “Towards a framework for more robust validation and verification of simulation models for systems of systems”. *The Journal of Defense Modeling and Simulation* vol. 13 (1), pp. 3–16.

Zeigler, B. P., J. J. Nutaro, and C. Seo. 2016. “Combining DEVS and Model-Checking: Concepts and Tools for Integrating Simulation and Analysis”. Int. J. Process Modeling and Simulation. I3M 2014: Special Issue on: "New Advances in Simulation and Process Modelling: Integrating New Technologies and Methodologies to Enlarge Simulation Capabilities.

Zeigler, B. P., J. J. Nutaro, and C. Seo. 2017. “Combining DEVS and model-checking: concepts and tools for integrating simulation and analysis”. *International Journal of Simulation and Process Modelling* vol. 12 (1), pp. 2–15.

Zeigler, B. P., and H. S. Sarjoughian. 2013. *Guide to Modeling and Simulation of Systems of Systems*. Simulation Foundations, Methods and Applications. Springer.

Zeigler, B. P., C. Seo, R. Coop, and D. Kim. 2013. “Creating Suites of Models with System Entity Structure: Global Warming Example”. In *Proc. of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium*, DEVS 13, pp. 32:1–32:8. San Diego, CA, USA, Society for Computer Simulation International.

Zeigler, B. P., C. Seo, and D. Kim. 2013. “System Entity Structures for Suites of Simulation Models”. *International Journal of Modeling, Simulation, and Scientific Computing* vol. 04 (03), pp. 1340006.

AUTHOR BIOGRAPHIES

LAURENT CAPOCCHI has been an assistant professor in computer sciences at the University of Corsica (France), SPE CNRS 6134 Research Lab since 2007. His email address is capocchi@univ-corse.fr.

JEAN-FRANCOIS SANTUCCI has been a full professor in computer sciences at the University of Corsica (France), SPE CNRS 6134 Research Lab since 1995. He was assistant professor at the EERIE School, Nimes, France from 1987 to 1995. His email address is santucci@univ-corse.fr.

BERNARD P. ZEIGLER is Chief Scientist at RTSync Corp., Professor Emeritus of Electrical and Computer Engineering at the University of Arizona, Tucson and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His email address is zeigler@rtsync.com.