Deterministic and Heuristic Criteria for Optimized Markov Chain Aggregation

Laurent Capocchi, Jean-François Santucci

▶ **To cite this version:**

Laurent Capocchi, Jean-François Santucci. Deterministic and Heuristic Criteria for Optimized Markov Chain Aggregation. 2024. ⟨hal-04840226⟩

HAL Id: hal-04840226

https://hal.science/hal-04840226v1

Preprint submitted on 17 Dec 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Noname manuscript No.
(will be inserted by the editor)

Deterministic and Heuristic Criteria for Optimized Markov Chain Aggregation

Laurent Capocchi · Jean-Fraçois Santucci

Received: date / Accepted: date

Abstract Markov chains are an important means of representing stochastic systems, but their state space can become too large to be practical. Markov model lumping is a well-known process for reducing the state space and simplifying interpretation, as well as accelerating the computationally expensive large Markov chain resolution. This paper investigates new Markov chain aggregation algorithms defined by deterministic (mathematically proven) and heuristic criteria using the Kullback-Leibler metric to compare Markov chains based on their respective steady states and the Mean First Passage Time property to optimize the state aggregation process. The objective is to strike a balance between the two conflicting objectives: (i) to minimize the size of the aggregation and (ii) to keep the aggregation as close as possible to the original Markov chain. A set of experiments has been conducted to perform illustrative comparisons that demonstrate the advantages gained from the proposed state-space reduction algorithms.

Keywords Complex systems · Dimensionality reduction · Heuristic algorithms · Iterative algorithms · Markov processes · Python

Mathematics Subject Classification (2020) 60J22 · 65C40 · 68M20 · 68T20

L. Capocchi
SPE UMR CNRS 6134
University of Corsica
Campus Grimaldi, 20250 Corte (France)
E-mail: capocchi@univ-corse.fr

J.F. Santucci
SPE UMR CNRS 6134
University of Corsica
Campus Grimaldi, 20250 Corte (France)
E-mail: santucci@univ-corse.fr

1 Introduction

Markov chains [22] are one of the simplest forms of stochastic dynamical systems that allow one to model stochastic dependencies. Beyond the ability to simulate a wide range of complex stochastic systems, the main interest of Markov chains is their ability to predict their behavior. The simulation of an aperiodic and irreducible Markov chain converges towards a limit stationary distribution, known as the steady state, which allows for state prediction. However, dealing with real-world complex system applications requires reducing the Markov chains due to their large state spaces, making it impossible to manipulate them.

An efficient solution for reducing the size of the state space of a Markov chain is to use aggregation [4]. Aggregation is typically based on defining a function to partition the nodes in the probability transition graph associated with the large-scale chain under study. Sets of nodes that have strong interactions are combined and considered as a single aggregated node in a new, lower-order Markov chain. The resulting new Markov chain should have dynamics that are similar to the original one, despite potential differences due to the state aggregations.

To reduce a Markov chain using aggregation, a set of approaches discussed below has been defined based on the reduction of the state set involving properties such as exact-lumpability and quasi-lumpability [40] and [22]. 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 exact aggregation and ensures that the process (chain) obtained after aggregation according to the partition remains a Markov chain. However, few chains have this property in practice, and it is helpful to use a lower threshold: quasi-lumpability. The proposed approach uses the exact-lumpability technique that guarantees accurate results, unlike quasi-lumpability.

Once a reduced Markov chain is generated from an initial Markov chain using lumpability techniques based on state partitions [7,41], the next step is to define a search methodology to select the optimal partition according to a metric that allows the comparison of Markov chains based on their respective steady states. This comparison helps to select the most promising reduced Markov chains and, therefore, reduce the potential differences between the original and the selected reduced Markov chain. One of the metrics associated with the lumpability process is the Kullback-Leibler (KL) divergence rate, as described in [19,18,32,3]. A search methodology is presented to select the optimal partition according to the KL divergence metric, which allows the comparison of Markov chains. An exhaustive comparison algorithm can be used to compute such an optimum. This algorithm involves computing the associated KL metric for each possible partition and returning the partitions having the lowest KL value. However, the computation process seems impossible for large Markov chains due to the CPU time consumption, as described in [19,13,18].

This article combines two different approaches to speed up computation processing and enable optimal partitioning according to the KL metric. The first approach simplifies the previous exhaustive algorithm by introducing a new deterministic criterion, which selects partitions with n-1 elements from the n-dimensional Markov chain. The second approach reduces the partition search space using a heuristic criterion based on a Markov chain property called the mean first passage time (MFPT) [10]. MFPT represents the time it takes to go from state i to state j for the first time. In [33], the authors have mathematically proven the relationship between MFPT and lumpability, implying that a semi-Markov process is lumpable if and only if the MFPTs are equal in distribution. The paper addresses a real optimization problem, which involves finding a balance between two conflicting objectives: (i) minimizing the size of aggregation using an exhaustive algorithm, and (ii) making the aggregation as similar as possible to the original chain using two algorithms that incorporate the previously mentioned improvements (deterministic and heuristic). In terms of the existing literature, the paper makes the following main contributions: (i) Presenting an exhaustive method with a complete algorithm that, given enough time and space, generates the optimal partition to reduce a stationary Markov chain. The advantage is that the optimal partition is found, but the disadvantage is that it is time-consuming and requires a lot of space. A benchmark on a large Markov chain illustrates the methodology and properties of the exhaustive algorithm (ii) Showing that the best partition (resulting in the minimum KL) belongs to the set of partitions obtained for the class k=n-1. The advantage is that this improvement is deterministic, but the disadvantage is that the resulting set of partitions is still too large (iii) Presenting an iterative algorithm to heuristically compute the MFPT property associated with the KL metric, which reduces the set of partitions obtained from the previous improvement. The advantage is that this approach significantly reduces the set of partitions, but the disadvantage is that the heuristic may not lead to the best reduction (iv) Introducing a new user-based heuristic algorithm (incorporating the previous algorithms) that allows finding the best partition that satisfies the two conflicting objectives presented earlier. The advantage is that a good partition is found, but the disadvantage is that it requires the user to make an effort to find the right heuristic (v) Providing examples that illustrate the presented method and the properties of solutions for both algorithms. A series of experiments have been conducted to make illustrative comparisons that demonstrate the advantages that can be gained from the application of the two improvement techniques proposed in this paper.

2 Background

2.1 Markov Chain Theory and Lumpability

In Markov theory, a Markov chain is a stochastic process $X(t)$ defined in a finite state space $S = 1, 2, \ldots, n$ [22] that generates a series of observations

X. The Markov property states that the probability distribution over the next observation depends only upon the current observation. Let $p_{i,j}$ be the probability that the next observation is j ($X(t+1) = j$) given that the current observation is i ($X(t) = i$). These transition probabilities are represented by a transition matrix P. In this paper, we only consider ergodic chains (i.e., the transition matrix must be irreducible and aperiodic) $M = (S, P, \pi)$. The advantages of ergodicity for the Markov chain aggregation model are highlighted by Deng in [14] when he explains that convergence of the aggregation algorithms is established based on stochastic approximation arguments as well as the ergodicity of the filtering process.

An ergodic Markov chain has a unique stationary distribution π (called the steady state) such that $\pi = \pi P$. However, when dealing with large Markov chains, manipulation can be difficult due to the large number of states. Lumpability is used to work at a higher level of probabilistic hierarchy using a partition of the set of states. Let us introduce two definitions to define quasi-lumpability and lumpability based on the probability matrix.

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

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

The classic definition of exact lumpability is obtained when $\epsilon = 0$.

Definition 2 If M is a Markov chain, S is the set of states, π is the steady state and P is 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 Markov chain $Q = \left(L, \widehat{P}, \widehat{\pi}\right)$ of order m, such that for all $i, j \in \{1, \ldots, m\}$ and for all $k \geq 0$ it holds,

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

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

The purpose of lumpability is to facilitate model reduction. Additionally, a metric is required to compare two Markov chains with different state spaces: N for the original Markov chain and M for the reduced Markov chain obtained after the partitioning process (with $dim(M) < dim(N)$). This process is based on a partition function ϕ that relates N to M. In this paper, we choose to use the KL divergence rate, which only considers chains of the same dimension, as defined below. We also provide the definition of the partition function.

Definition 3 For two stationary Markov chains (N, π, P) and (N, π', P') defined in the same state space N, the KL divergence rate $R(P||P')$ is a measure of the difference between these two Markov chains and is given by the following formula inspired by [28]:

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

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

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 follows.

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

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

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

We will illustrate the manipulation of Markov chains using a weather forecasting example. Figure 1 shows the probabilistic finite-state automata corresponding to a transition matrix with three states: *Sunny*, *Rainy*, and *Cloudy*. When the weather is *Sunny* (indicated by rst_n in Figure 1), the probability of having rain on the next day is 0.1, 0.9 for clouds, and 0.0 for remaining sunny. The same applies to the other two states.

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 calculated by solving 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. To illustrate the lumping process, we choose a partition $L = \{\{Sunny\}\{Rainy, Cloudy\}\}$ and the corresponding lumped Markov chain $Q = \left(L, \widehat{P}, \widehat{\pi}\right)$ obtained from Equation 1:

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

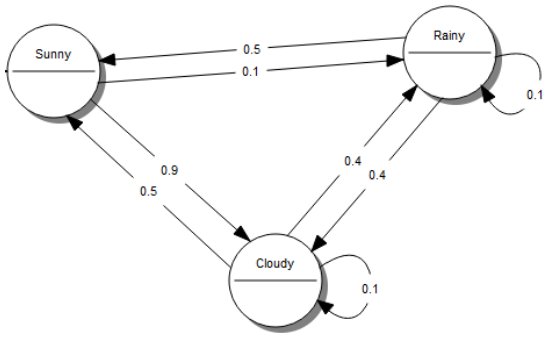

Fig. 1: Probabilistic finite state automata of the weather forecasting example with three states: Sunny, Rainy and Cloudy.

2.2 Mean First Passage Time and Kullback–Leibler

If an ergodic Markov chain P starts in state s_i, the expected number of steps to reach state s_j for the first time is called the Mean First Passage Time (MFPT) from s_i to s_j. It is denoted by m_{ij} and is calculated by the following equation:

$$m_{ij} = 1 + \sum_{k \neq j} p_{ik} m_{kj}, \tag{4}$$

where p_{ik} is an element of the transition matrix associated with the ergodic Markov chain.

It has been pointed out in [22] that the MFPT has the following property (P1): If a finite Markov chain with N states is lumpable with respect to a given partition $L = \{L(0), L(1), ..., L(M)\}$ of N, then its MFPT from $i \in L(m)$ to any state in $L(n)$ is the same for all $i \in L(m)$ for $0 \leq m, n \leq M$, $m \neq n$. These relationships between MFPT and lumpability have guided the definition of a heuristic criterion to guide the search for the optimal partition.

The problem of aggregating Markov chains can be summarized as follows: given a stochastic matrix of transition probabilities, the goal is to partition this matrix and produce a reduced-size matrix with fewer states. There are many possible aggregated stochastic matrices that can be formed for a given Markov chain, and the objective is to find a matrix that produces the least difference for a chosen measure, such as the Kullback-Leibler (KL) divergence. The KL divergence is widely recognized as a measure of change between Markov chains, which enables one to identify optimal properties [28]. In mathematical terms, this can be expressed as follows:

Let (N, P, π) be a given stationary Markov chain. The partition problem is to find the partition function $\phi : N \rightarrow M$ and the optimal aggregated Markov chain (M, Q, ω) (with $\dim(M) < \dim(N)$) such that $R^{(\phi)}\left(P||\widehat{Q}\right)$ (Equation 2) with $\widehat{Q}$ (Equation 3) is minimized:

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

The proposed approach is based on a combination of these two properties (MFTP and KL) to implement new Markov chain reduction algorithms.

3 Optimal Markov Chain Aggregation

3.1 Problem Statement

Basically, the reduction of the Markov chain can be seen as an optimization problem that aims to find the best aggregation of an n-dimensional Markov chain. This involves finding a balance between two conflicting objectives: (i) computing the smallest possible aggregation reflected in a pure optimization problem and (ii) determining an aggregation that is as similar as possible to the original chain in order to preserve its global dynamical properties (i.e., searching for lower KL values).

A first classical approach to finding the smallest possible aggregation is using an exhaustive algorithm to solve equation 5. To accelerate the algorithm, two types of algorithms can be used: (i) heuristic-based, which guides to an optimal solution on average, and (ii) deterministic-based, which always leads to an optimal solution. Both types of algorithms have advantages in terms of computation time. While deterministic algorithms have improved features such as CPU time and memory, heuristics may not always result in optimal performance. However, optimal heuristics are essential for efficiently performing the optimal partition generation process and complement deterministic improvements. These algorithms enable finding the optimal reduced Markov chain with n-1 states when the initial chain had n. However, the main objective is to find the optimal aggregation of an n-dimensional Markov chain. The proposed paper presents a new algorithm that loops over each reduced Markov chain obtained by the exhaustive algorithm, including deterministic and heuristic improvements, to find the optimal aggregated Markov chain which as similar as possible to the original chain. This means for this new algorithm that it seeks the minimum between the quality of the approximation and the number of aggregates.

The next subsection presents the exhaustive algorithm with deterministic improvement, and the following subsection describes the chosen heuristic and the corresponding improvement for generating an optimal aggregated Markov chain. The third subsection presents the new algorithm, which combines the previous algorithms to achieve the main objective.

3.2 Exhaustive Algorithm with Deterministic Improvement

The approach consists of solving the equation 5. The implementation is based on a function FOP (Find-Optimum Partition) given in Algorithm 1. After

reading the ergodic Markov chain $M = (S, P, \pi)$, a preprocessing phase is performed, which consists of generating the set of all possible partitions of states associated with M. Let us call LP^M the set of all partitions of a given Markov chain M. $FOP(LP^M)$ is a function whose parameter is the list of partitions that will be considered to find the "best" partition (not optimal but "best" due to the fact that LP^M contains all the possible partitions). For each partition p of LP^M (line 2 of Algorithm 1), the following three steps are executed:

- The m-dimensional Markov chain Q (where m is the number of macrostates corresponding to partition p) is computed using the function $Lump$ (line 3 in Algorithm 1). $Lump$ is implemented according to Equation 1.
- The lifted Markov chain $\widehat{Q}$ is obtained (line 4 in Algorithm 1) by performing the π-lifting of Q implemented in the function $Lifting$ based on Equation 3.
- The KL value R corresponding to the KL divergence rate between M and $\widehat{Q}$ is calculated using the function KL based on Equation 2 (line 5 in Algorithm 1).
- The tuple (R, P, p) is added to the list LP^M (line 6 in Algorithm 1), which will be analyzed by the function $argmin_R$ to find the optimal partition p' associated with the optimal KL value R' and the lump transition matrix P' that are encapsulated in the set (R', P', p') (line 8 in Algorithm 1).

Algorithm 1 Algorithm for the Find-Optimal-Partition (FOP) function that returns the optimal partition according to a given list LP^M.

Input: LP,π,S,P
Output: (R',P',p')
1: $FOP(LP^M)$
Initialization
2: $LR \leftarrow$ Null
Loop Process
3: **for all** p in LP^M **do**
4: $Q \leftarrow Lump\,(p, \pi)$
5: $\widehat{Q} \leftarrow Lifting\,(Q, p, \pi, S)$
6: $R \leftarrow KL\left(\widehat{Q}, \pi, S, P\right)$
7: $LR \leftarrow (R, P, p)$
8: **end for**
9: **return** $(R', P', p') \leftarrow \arg\min_R LR$

Then, the best partition p' (associated with its KL value R and the transition matrix P') is obtained by selecting the corresponding minimum value of R according to LR (line 8 in Algorithm 1). However, the computation can only be performed on small state spaces. Indeed, according to Algorithm 1, which contains a loop (line 2), its time complexity is considered to be $O(n^3)$, where n is the number of partitions (the length of the list LP^M) to be parsed. Obviously, the larger the state space (large Markov chain), the greater the time complexity (CPU time and memory space). We therefore propose an

improvement to the exhaustive algorithm based on deterministic criteria. By running the previous Algorithm 1 on examples with small state spaces, we noticed that the best solutions belonged to partitions with $n-1$ elements of the n-order Markov chain. Given the nature of the lifted matrix $\widehat{Q}$, this leads to a theorem $P(n)$ mathematically demonstrated in [8]. The Perron-Frobenius [27] reduction method also uses this theorem.

The use of theorem $P(n)$ allows us to considerably reduce the number of partitions to be considered by the exhaustive algorithm of the previous section. Instead of considering all partitions (LP^M) associated with an n-ordered ergodic Markov chain of order n M, using the theorem $P(n)$, we can just consider a subset of LP^M corresponding to partitions that have $n-1$ elements (called $LP^M n-1$). Then, the optimal partition can be obtained by performing $FOP(LP^M n-1)$.

Even though the number of partitions is drastically reduced when searching for the optimum using the FOP function, the methodology still cannot find the optimum for large Markov chains in an acceptable CPU time. We therefore propose a heuristic criterion that can guide the search towards the most promising partitions.

3.3 Exhaustive Algorithm with Deterministic and Heuristic Improvements

To guide the selection of the set of potential partitions (LP^M_{n-1}) in the search for the optimal solution, the notion of MFTP can be associated with the concept of lumpability of a Markov chain (as explained in Section 2). This is a novel idea that has not been explored in previous scientific work related to determining the best aggregation solution (as discussed in Section 5).

Since there is a strong link (P1 property defined in Section 2) between lumpability and the MFPT of the Markov chain states, a heuristic criterion has been defined to propose an ordering between sets of potential partitions.

This heuristic criterion can be used to reduce the list of potential partitions before applying Algorithm 1. The reduction is done to avoid trying all partitions, as some partitions pointed out by the MFPT analysis can be overlooked.

The heuristic criterion is defined as follows: For each partition p, the algorithm computes the minimum of the differences in $MFPT(p)$. This produces a list of values that reflects the lumpability of a given partition p. Selecting the minimum value of the set of MFPT values as the heuristic criterion gives priority to partitions that have a greater ability to satisfy property P1 (and reflect the potential lumpability of the initial ergodic Markov chain M with respect to partition p).

The implementation of the previous heuristic, which allows one to compute the reduced list of partitions to be considered (called $NewLP$), is given in Algorithm 2. The function $REDUCED$ is based on the heuristic criterion to reduce the number of partitions to be considered as a potential optimal

solution. The optimal partition is then obtained by performing $FOP\left(REDUCED\left(LP_{n-1}^{M}\right)\right)$.

Algorithm 2 Algorithm of the function $REDUCED$ that uses the MFPT-based heuristic to reduce a given list of partitions LP^M.

Input: LP^M
Output: $NewLP$
1: $REDUCED(LP^M)$
Initialization
2: $NewLP \leftarrow$ Null
Loop Process
3: **for all** p in LP^M **do**
4: $H_p \leftarrow (p, MFTP(p))$
5: **end for**
6: $Mean \leftarrow MEAN(H_p)$
7: **for all** (p, h) in H_p **do**
8: **if** $h < Mean$ **then**
9: $NewLP \leftarrow p$
10: **end if**
11: **end for**
12: **return** $NewLP$

In the loop of Algorithm 2 (line 2), p contains m elements corresponding to the m macro-states. Let us call these elements p_i where $1 \leq i \leq m$. The function $MFPT$ computes (line 3) the minimum of the difference between the MFPT of state p_i and each state belonging to p_j where $i \neq j$. Let us call this value H_p. Line 5 of Algorithm 2 shows the call to the $Mean$ function, which computes the mean of the MFPT values. The proposed heuristic is based on the mean of the MFPT values. The list $newLP$ is obtained by reducing the list H_p according to the value of $Mean$ (lines 6-10).

The MFPT-based heuristic generally leads to a set of partitions that includes the optimal one when using the $Mean$ function. However, using this heuristic has some disadvantages, since an optimal partition cannot always belong to the final set of proposed partitions.

Despite the minor deficiencies mentioned above, we expect that the introduction of the MFPT heuristic will accelerate the overall optimal search process and further improve the performance of the exhaustive algorithm, as pointed out in Section 4.

3.4 New Optimal Aggregation Algorithm

In the two previous subsections, we presented algorithms that allow us to find the optimal Markov chain reduction to $n-1$ states when the initial chain had n states. However, the main goal is to find the optimal aggregation of an n-dimensional Markov chain (the original chain) with as few aggregated states as possible while being as similar as possible to the original chain.

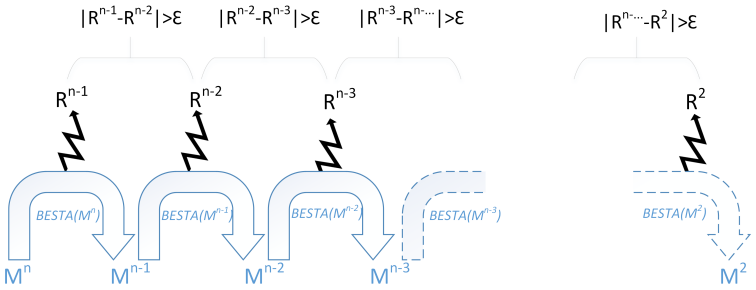

Fig. 2: Illustration of Algorithm 3 where the aggregation process is executed from an original n-dimensional Markov chain M^n to the 2-dimensional one M^2. At each step, the Markov chain is reduced by one state and a new KL divergence (noted R) is generated from the $BESTA(M)$ function. The stopping condition may depend on the value of the ϵ parameter.

An algorithm based on the FOP function presented in Algorithm 1 is proposed in Figure 2. At each step i, the optimal partition algorithm for $n-i$ is executed to produce the lumped Markov chain M^{n-i}, and the error limits are computed using the values R^{n-i}. A stopping strategy is based on a heuristic that allows us to determine when the algorithm is complete, resulting in the Markov chain with the smallest possible aggregated states while still being similar to the original chain (similarity is measured using the KL values).

Algorithm 3 Algorithm for finding the optimal aggregated transition matrix P' for a Markov chain $M' = (S', P', \pi')$ with respect to the minimum number of states and the maximum similarity to the initial Markov chain $M = (S, P, \pi)$.

Input: M
Output: P'
1: $BESTA(M)$
Initialization
2: $R, P', p' \leftarrow FOP(REDUCED(LP^{M}_{n-1}))$
3: $M' \leftarrow (STATES(P'), P', STEADY(P'))$
4: $NewR \leftarrow 0$
5: $n \leftarrow dim(M)$
Loop Process
6: **while** *condition* **do**
7: $NewR, P', p' \leftarrow FOP(REDUCED(LP^{M'}_{n-1}))$
8: $M' \leftarrow (STATES(P'), P', STEADY(P'))$
9: $R \leftarrow NewR$
10: $n \leftarrow n-1$
11: **end while**
12: **return** P'

The K-L divergence rate acts as a limit to the reduction error of the model, setting the error bounds. Algorithm 3 works as follows: the functions $STATES(P')$ and $STEADY(P')$ (line 2) compute, respectively, the list of states S and the steady-state value π of a transition matrix P', which are used to define the reduced Markov chain M'. $NewR$ and n are initialized to zero and the number of states of the initial Markov chain M (lines 3 and 4), and are updated during the loop process controlled by a condition statement in a while loop (line 5). The new KL divergence ($NewR$), the new transition matrix P', and the best partition p' are calculated by executing the function FOP on the result of the $REDUCED$ one (line 6). The current Markov chain M', the KL value R, and the variable n are updated from lines 7-9.

The stopping condition (line 5) can be expressed as the difference between the current KL divergence ($NewR$) and the previous one (R), which is compared to a parameter ϵ ($NewR - R > 0$ and $NewR - R > \epsilon$). In this way, the stopping condition depends on the deviation of the KL divergences and can be defined as a percentage. Another way is to define the stopping condition in relation to the desired size of an aggregated Markov chain with dimensions N, by implementing the rule $n \geq N$. However, if the stopping condition is never reached, the algorithm ends with a minimally aggregated two-dimensional Markov chain P'. Since the $FOP(REDUCED(LP^{M'}_{n-i}))$ function (with $i \in [2, dim(M')]$) gives the best (n-i)-dimensional aggregated Markov chain (i.e., with the best KL divergence compared to the previous n-dimensional Markov chain), the optimal aggregated Markov chain can be obtained when the stopping condition is reached.

In fact, finding the optimal aggregation of an n-dimensional Markov chain, with the stopping condition based on ϵ, having as small as possible aggregated states and being as similar as possible to the original chain, consists of finding the appropriate ϵ, which can be defined as a tracking error that may not exceed a specific value (expressed as a percentage, for example). When the stopping condition is specified by a specific number of states (dimension) of the expected aggregated Markov chain to reach, the control is easier to define.

3.5 Time Computational Complexity Formal Analysis

The algorithm 1 is evidently intractable and not very practical because its time complexity (both in the worst case and in the average case) is not polynomial, in general.

According to Equations 1, 3, and 2, the total execution time of algorithm 1 is $N * (NNc_1 + NNc_2 + NNc_3)$, with c_1, c_2, and c_3 being the constant times corresponding to the execution of the statements called in the Lump, Lifting, and KL functions. The time complexity is considered to be $T(N) = O(N^3)$, where N is the number of partitions (the length of the list LP^M) to be parsed.

The total execution time of algorithm 2 is $N*t+c+N$, with t being the time corresponding to the execution of the function MFPT (line 2) that depends on the partition p and is equal to $t = len(p)c$, where c is the constant time

corresponding to the multiplication operation involved in Equation 4 and p is the Markov chain of LP^M. Now ignoring the lower-order terms, since they are relatively insignificant for large input $Nlen(p)$, only the highest-order term is taken (without constant), which is N in this case. The time complexity is considered to be $T(N, len(p)) = O(N)$, because $len(p)$ is relatively insignificant compared to N, which is the number of partitions obtained from p. During the analysis of the algorithm, mostly the worst-case scenario is considered, i.e., for a large Markov chain.

The time complexity of algorithm 3 is $T(N) = O(N^3 + Z * N^3) = O(N^3(1 + Z))$, because the function $FOP(REDUCED(LP^M_{N-1}))$ has a time complexity equal to $T(N) = O(N^3)$. The variable $Z \in [0, dim(M) - 1]$ is related to the while condition (called on line 5) and can possibly be neglected in front of N^3. The order of growth is how the execution time depends on the length of the input. It is evident that the execution time cubically depends on the number of partitions N, which is lowered by the fact that we choose to apply the BESTA algorithm on the LP^M calculated for the class $k = N - 1$ at each step of the while condition driven by the statement *cond*. The worst-case scenario is considered when *cond* is true until reaching the maximum steps of the while loop ($Z = dim(M) - 1$), while the best case is considered when *cond* is false ($Z = 0$). In terms of comparison, the computational time complexity of the Partial Sum and the Gerschgorin functions is $O(N^2)$, and the Markov Chain Indexing method is $O(N^4)$.

Regarding the relations between the error bounds encountered and the computational time complexity, the more you reduce, the longer the computation time, and the greater the error. That is why the stopping condition allows us to obtain a result fairly quickly without too much error.

4 Experiments and Results Analysis

Algorithms and test benches, including Markov chains ranging from 3 to 1000 states (randomly generated), have been implemented in the Python language (v. 3.9.6) and executed on a Windows 10 operating system with an Intel (R) Xeon (R) (E-2176M CPU @ 2.70GHz, 2712 MHz, 6 cores) processor and 32GB of RAM. All the code used to obtain the results presented in this paper is available in [8].

4.1 Validation of Deterministic and Heuristic Based Improvements

To validate the benefits of using deterministic and heuristic improvements, we propose conducting two types of experiments based on ergodic Markov chains normalized to a size of n (number of states) and a chosen distribution drift (uniform, Rayleigh, binomial, Weibull, and beta) [1]. Stochastic matrices will be generated using random numbers, subject to certain conditions:

- The rows must sum up to 1.

– The values on the diagonal should be significantly higher than the other values.

The first experiment compares the exhaustive algorithm with a deterministic improvement based on 11 randomly uniformly distributed Markov chains from the benchmark. It is not necessary to consider large Markov chains to validate the deterministic improvement.

On the other hand, validating the heuristic algorithm requires considering a large Markov chain to demonstrate the benefits of the reduction rate (i.e., the number of partitions the heuristic algorithm needs to consider to find the optimal partition relative to the total number of partitions).

Table 1: The efficiency of both the deterministic and heuristic improvements was validated on large, normalized, ergodic Markov chains that were uniformly distributed with high probability, where the parameter *high* was set to 0.1.

Matrix (n x n)	Nb. of partitions	Nb. of partitions for k=n-1	Time for k=n-1 [s]	Best KL	Optimal partition max length	FOP [s]	REDUCED [s]	FOP(REDUCED) [s]	Reduction Rate (compared to k=n-1) [%]
3x3	5	3	1.091	0.0927	2	1.0987	$6.9e^{-7}$	0.0028	**33.33**
4x4	14	5	1.1018	0.1679	3	1.1106	$6.9e^{-7}$	0.0043	**40.00**
5x5	49	8	1.0830	0.1403	7	1.0815	$8.0e^{-7}$	0.0083	**12.50**
6x6	198	13	1.1003	0.0829	2	1.1545	$7.0e^{-7}$	0.014	**7.69**
7x7	869	17	1.0891	0.0732	14	1.465	$6.0e^{-7}$	0.020	**17.64**
8x8	4130	24	1.1319	0.0694	18	3.295	$8.0e^{-7}$	0.046	**25.00**
9x9	2110	32	1.1628	0.0622	24	14.974	$7.0e^{-7}$	0.053	**25.00**
10x10	11600	40	1.2344	0.0524	30	93.988	$8.0e^{-7}$	0.092	**25.00**
15x15	$1.38e^{+09}$	99	1.6420	0.037	68	707.23	$8.0e^{-7}$	0.390	**31.31**
20x20	$5.17e^{+13}$	181	2.9050	0.0279	138	4746.9	$8.9e^{-7}$	1.485	**23.75**
30x30	$8.47e^{+23}$	422	9.9782	0.0148	259	35588	$8.0e^{-7}$	6.777	**38.62**
40x40	$1.57e^{+35}$	762	31.6959	0.0115	496	$> 10e^{3}$	$1.5e^{-6}$	19.526	**34.90**
50x50	$1.86e^{+47}$	1200	77.3586	0.0077	799	-	$8.0e^{-7}$	45.942	**33.41**
60x60	$9.77e^{+59}$	1740	165.386	0.0046	1192	-	$9.0e^{-7}$	108.486	**31.49**
70x70	$1.81e^{+73}$	2380	312.5046	0.004	1508	-	$8.9e^{-7}$	220.963	**36.63**
80x80	$9.91e^{+86}$	3120	551.0799	0.0036	2049	-	$8.9e^{-7}$	361.596	**34.32**
90x90	$1.42e^{+101}$	3960	883.8726	0.003	2628	-	$1.0e^{-6}$	558.582	**33.63**
100x100	$4.76e^{+115}$	4900	1391	0.004	3214	-	$9.9e^{-7}$	962.731	**34.40**
1000x1000	$5.56e^{+401}$	15760	$> 10e^{8}$	$< 10e^{-5}$	6178	-	$< 10e^{-8}$	$> 10e^{3}$	**12.56**

The results are summarized in Table 1. They demonstrate the efficiency of deterministic and heuristic improvements and can be reproduced by executing the script *table1.py*, which is available from [8].

Table 1 shows the results for 19 n-ordered Markov chains. Specifically, it displays the total number of partitions, the number of partitions considered when computing the optimum one (when $k = n-1$), the CPU time required to perform the KL computation and obtain the corresponding optimal partition when $k = n - 1$, the obtained value for the best KL, the maximum length for the reduced partitions set using the heuristic algorithm based on MFPT, the CPU time to find the best partition from all possible partitions (FOP), the CPU time to reduce the list of all partitions using the MFPT-based heuristic, the CPU time required to perform the KL computation and obtain the optimal partition on the set of reduced partitions using the MFPT-based heuristic, and the reduction rate.

The reduction rate expresses the interest in heuristic improvement by computing 1 minus the length of the partition set obtained after applying the

heuristic process on the number of partitions to be considered by the deterministic process ($k = n - 1$). This reduction rate expresses the impact in terms of the number of partitions to be considered when computing the best KL value and the corresponding optimal partition.

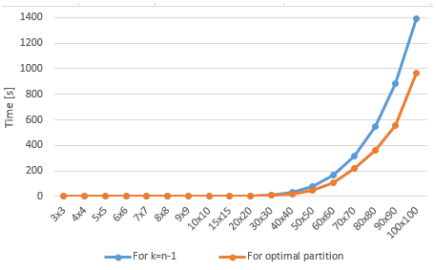

Fig. 3: CPU times are required to perform the KL computation and obtain the corresponding optimal partition when KL is minimum with k=n-1 (labeled "For k=n-1" and corresponding to the column "Time for k=n-1" in Table 1), as well as the CPU times required to perform the KL computation and obtain the optimal partition on the reduced partition set using MFPT heuristic (labeled "For optimal partition" and corresponding to the column "$FOP(REDUCED)$" in Table 1).

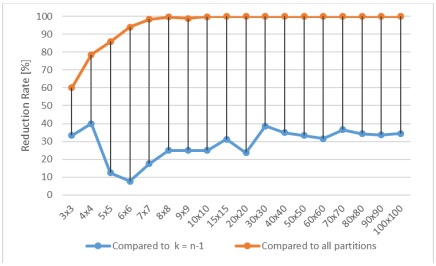

Fig. 4: The number of partitions to be considered by the heuristic algorithm to find the optimal partition is compared for two scenarios: when the number of partitions is set to k=n-1, labeled 'Compared to k=n-1', and when it is compared to the total number of partitions, labeled 'Compared to all partitions'.

From Table 1, it becomes evident that using both the deterministic and heuristic improvements, the computation of the optimal partition can be performed on large Markov chains. Without these improvements, it is not possible to compute the search for the optimal partition for large Markov chains, as highlighted in the 7th column "$FOP(LP)$". Furthermore, the CPU time to compute the KL metrics and the optimal partition is drastically reduced (see the fourth and ninth columns) as presented in Figure 3. One can also notice that these CPU times appear to be small in comparison to the CPU times required for performing the KL computation and the optimal partition without the heuristic improvement.

We also point out that the best KL is found for each Markov chain belonging to the set of selected partitions using the heuristic. So, in our benchmark, the heuristic criterion allows finding the best partition (this may not always be true). Finally, the last column of Table 1 confirms that the search for the

optimal partition algorithm performs much better when using the heuristic improvement (see "Compared to k=n-1" in Figure 4).

In addition, it is quite clear that the heuristic benefit is also highlighted by comparing the three columns that deal with the number of partitions to be considered in the algorithms. The curve "Compared to all partitions" in Figure 4 shows that 100% is reached for large space Markov chains, which highlights the benefit of the heuristic process.

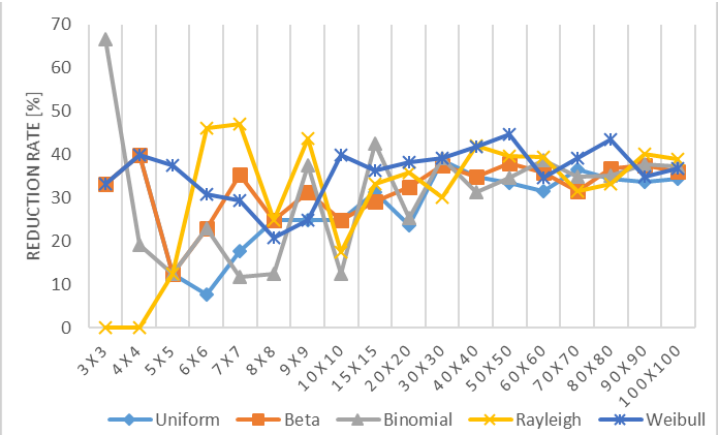

Fig. 5: The reduction rate compared to k=n-1 can be expressed as a function of the distribution used to randomly generate the initial Markov chain.

Furthermore, to demonstrate the impact of the heuristic process, we performed an additional experiment using normalized initial ergodic Markov chains generated using five different distribution drifts. The results of this experiment are summarized in Figure 5, which shows the reduction rates obtained by comparing the number of partitions obtained after the heuristic process with the number of partitions for $k = n - 1$.

Figure 5 confirms that the heuristic process results in a significant reduction in the dimensionality of the number of partitions to be considered. For all distributions, the reduction rate is between 30 and 40 when the transition matrices are large, which is a good result.

4.2 Validation of the Optimal Aggregation Algorithm

Two types of experiments are proposed to validate the BESTA algorithm (Algorithm 3) for optimal aggregation. The first experiment is based on a 16-dimensional Markov chain, which allows us to demonstrate the proposed aggregation at each step of the algorithm and show how the two conflicting objectives can be taken into account.

The second experiment aims to validate that the proposed algorithm results in an aggregated Markov chain that is still semantically correct. To achieve this, a well-known case study on weather forecasting has been extended and used as an example.

4.2.1 Conflicting Objectives Problem Illustrative Example

This subsection aims to validate the use of the BESTA algorithm. Figure 6 illustrates an initial Markov chain involving 16 states and demonstrates the use of the parameter ϵ to break the loop involved in the BESTA algorithm (line 5 in Algorithm 3) to obtain the optimal aggregated Markov chain when the standard deviation of the KL divergence should not exceed 50%. As per the notation introduced in Section 3.4, the stopping condition is $|newR - R| < \epsilon$, with $\epsilon = R * 0.5$.

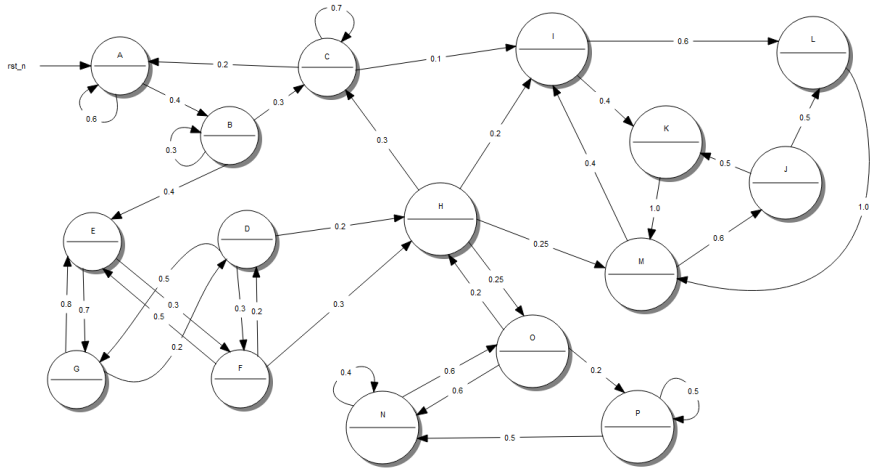

Fig. 6: Transition graph of the 16x16 Markov chain involved in the conflicting objectives problem illustrative example.

Table 2 presents the results obtained, which can be reproduced by executing the *table2.py* script available at [8]. The computation of the KL at each step of the BESTA algorithm is given in the column "Best KL," while the difference between two consecutive KLs is given in the column "Best KL Difference." The last column shows the differences between the initial 16-dimensional Markov chain and the aggregated one. The results show that the 10x10 transition matrix presents the best compromise between the conflicting objectives described above, as ϵ becomes smaller than the corresponding KL difference when using the 10x10 transition matrix (as shown in bold in Table 2).

To validate that the 10x10 matrix obtained with $\epsilon = 50\%$ is the best compromise, we propose to calculate the KL differences between the initial

Table 2: Validation of the BESTA algorithm applied on the 16-dimensional Markov chain. The optimal reduced Markov chain is obtained as soon as the ϵ value is less than the corresponding KL difference.

Matrix (n x n)	Best KL (e^{-3})	KL Difference $\lvert newR - R\rvert$ (e^{-3})	$\epsilon = R * 0.5$ (e^{-3})	KL Difference against 16x16 matrix (e^{-3})	Time [s] (e^{-3})
16x16	1.2451				1395.2
15x15	2.2361	0.991	0.622	2.8	783.7
14x14	2.3985	0.162	1.118	4.9	340.8
13x13	2.7820	0.383	1.199	7.1	315.1
12x12	2.0978	0.684	1.391	10.0	223.6
11x11	2.9157	0.817	1.0489	11.3	0.1571
10x10	**11.5080**	**8.592**	**1.457**	**13.7**	**102.4**
9x9	29.8988	18.390	5.754	67.6	71.3
8x8	21.0032	8.895	14.9494	77.0	48.9
7x7	33.6406	12.637	10.5016	759.7	24.3
6x6	29.6560	3.984	16.8203	790.5	18.0
5x5	263.8485	234.192	14.828	919.5	7.6
4x4	573.4026	309.554	131.92425	1292.5	6.0
3x3	584.1693	1.076	286.7013	1384.6	0.03

16x16 matrix and all reduced matrices (from 15 to 3 states) as presented in the fifth column of Table 2. The last column indicates the computational time complexity of the BESTA algorithm. The first value gives the computational time spent by the first phase of the BESTA algorithm 3 (lines 1-4), which takes 0.6527 s plus the first pass of the while loop (to reduce the 16x16 matrix to the 15x15 matrix). The other values of the last column (from n=15 to 3) give the computational time spent for the corresponding pass of the while loop only. For example, for n = 10, the time spent to calculate the reduction of the initial chain is the sum of all values from line 1 to line 7. Obviously, as explained in Section 3.5, the time and error bound increase when performing the BESTA algorithm.

We can observe from the fifth column of Table 2 that the best matrix, in terms of the number of states and its similarity to the initial matrix (measured by the KL divergence), is the 10x10 matrix. The KL difference between the 10x10 matrix and the initial matrix is close to that of the 11x11 matrix, and it is also the last reduced matrix that has the lowest KL difference compared to the values obtained from the 9x9 matrix to the 3x3 matrix.

The error bound (K-L divergence rate in column 3 of Table 2) is a function of the number of aggregated states (column 1 of Table 2). The error bound indicates that $R^{(\phi)}\left(P||\widehat{Q}^n\right)$ (the best KL given in Equation 2) increases slowly from n=15 to n=10. Due to the ϵ-based stopping condition, the best partition is obviously the one that corresponds to k=n-1. Of course, we reject this obvious solution and let the algorithm find the best compromise between reducing the number of states and losing information.

4.2.2 Semantic Conservation.

Let's extend the example of weather forecasting presented at the end of Section 2 by adding six new states. The state *Sunny* is now renamed to Sunny-with-wind and is associated with two additional states: Sunny-without-wind and Sunny-heat-wave. Similarly, the state *Rainy* is renamed to Rainy-with-hail and is expanded with two new states: Rainy-with-wind and Rainy-with-thunder. Finally, the original state *Cloudy* is expressed more specifically using three states: Cloudy-altostratus, Cloudy-stratus, and Cloudy-cumulus. Figure 7 shows the transition graph of this new Markov chain, with probabilities defined to maintain a similar ratio to the original (Figure 1).

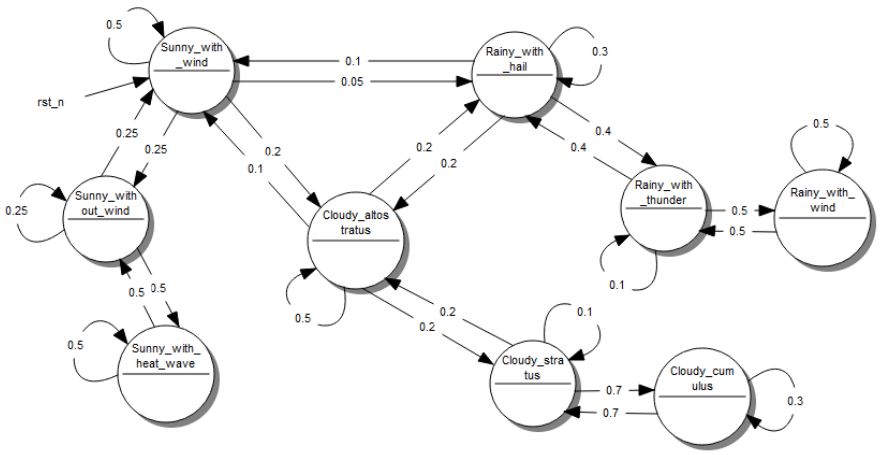

Fig. 7: Transition graph of the previous weather forecasting example with 6 new additional states.

The goal is to show how the BESTA algorithm is able to propose the optimal aggregation scheme ($L = \{\{Sunny\}\{Rainy, Cloudy\}\}$) according to the extension of the Markov chain of weather forecasts presented in Section 2. Figure 8 shows the KL values for each step of the execution of the BESTA algorithm on the 9-dimensional Markov chain of extended weather forecasts. The KL value increases until the reduced 3-dimensional Markov chain, which has been chosen as a stopping condition of the BESTA algorithm. Figure 9 confirms the effectiveness of the BESTA algorithm. Each state graph is presented to observe the semantic effect of the proposed reduction process. Figure 9 (f) shows the 3-dimensional Markov chain that predicts weather where states *Cloudy* and *Rainy* are quasi-merged in front of the group of states *Sunny*.

The lumping quality measure is taken as the error in the steady-state distribution of the aggregated matrix relative to that of the initial matrix. Indeed, the last reduced Markov chain is composed of three states: (Sunny-with-heat-

wave / Sunny-without-wind / Sunny-with-wind), (Cloudy-altostratus / Rainy-with-hail / Cloudy-cumulus / Cloudy-stratus), (Rainy-with-thunder / Rainy-with-wind). The corresponding steady state is $\pi = [0.400, 0.413, 0.186]$, which is close to the previous steady state obtained in Section 2 $\pi = [0.3333, 0.6666]$ for partition $L = \{\{Sunny\}, \{Rainy, Cloudy\}\}$. The reduced Markov chain must preserve the stationary distribution of the original Markov chain. The reduced model is easier to interpret since it describes the dynamics between understandable and stable groups of states. The probability that today is cloudy or rainy is 0.666 in the initial weather forecast example (not extended), compared to $0.413 + 0.186 = 0.599$ with the reduced extended version presented in this section. Similarly, the probability that today is sunny is 0.333 in the not-extended version, compared to 0.400 in the extended one.

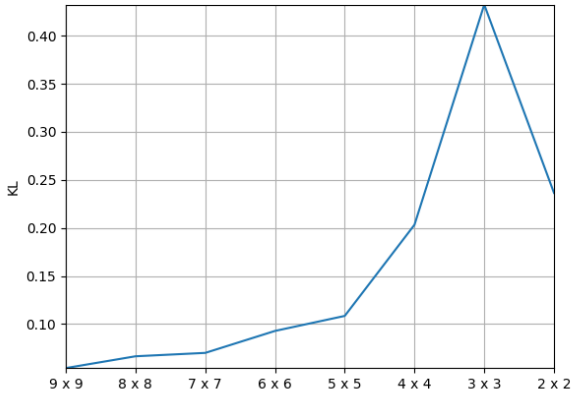

Fig. 8: KL values during the BESTA algorithm execution on the 9-dimensional extended weather forecasting Markov chain.

4.3 Discussion and Comparison

We specifically study the case of a continuous-time M/M/1 queue [6] for discussion and comparison. The term M/M/1 comes from Kendall's notation: the two M's specify the probabilistic law governing arrivals and services (in this case, a Markovian process), and the 1 specifies that only one server processes the queue. The process described above is represented by the following stochastic matrix P (ergodic):

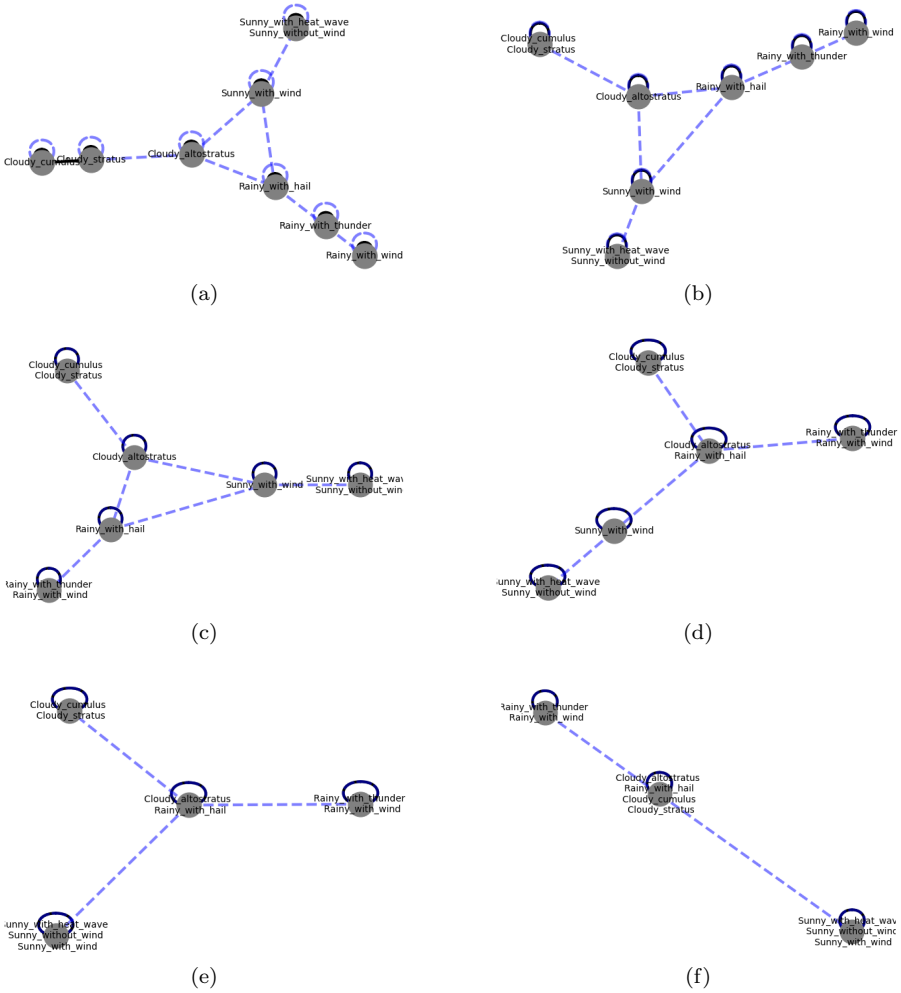

Fig. 9: State Graphs of Markov chains resulting of the BESTA function execution from the weather forecasting 9-dimensional Markov chain for the first step (a) to the last one (f). At each step, the aggregation of two states gives the optimal reduced Markov chain. For example, the states *Sunny_with_heat_wave* and *Sunny_without_wind* are aggregated at the end of the 1st step (a). The solid line between the states *Cloudy_cumulus* and *Cloudy_stratus* denotes a probability value greater than 50%.

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \cdots \\ \lambda & -(\lambda+\mu) & \mu & 0 \cdots \\ 0 & \lambda & -(\lambda+\mu) & \mu \cdots \\ \vdots & \vdots & \vdots & \vdots \ddots \end{bmatrix},$$

where the symbols λ and μ are used to represent the arrival and service rates, respectively, in an M/M/1 queue. The service rate, is the rate at which customers are served by the server and it determines the speed at which customers are processed and the overall performance of the system. However, the other metrics also play important roles in understanding the behavior of the M/M/1 system. The waiting time is the time a customer spends waiting in the queue before being served, and is affected by both the arrival rate and the service rate. The service time is the time required to serve a customer and is determined by the service rate. It is through the respect of the previous parameters that we will address the issue of defining the stopping condition of the BESTA algorithm. We will also rely on these parameters to validate our approach against three traditional methods for reducing P: Partial Sum, Gerschgorin, and Markov Chain Indexing methods that are mentioned in the Section 5.

4.3.1 Benchmark with Common Methods

To show why the lumping algorithm may be more effective than other reduction algorithms for Markov chains, we compare the BESTA algorithm with the following alternative reduction methods: (i) Partial Sum with $1e^{-5}$ for the tolerance for convergence as usually defined (ii) Markov Chain Indexing Method and (iii) Gerschgorin.

We generated a 20-dimensional M/M/1 Markov matrix with service rate of 40 and arrival rate of 20. It is not necessary to generate a larger matrix, as we do not aim to demonstrate the ability of our approach to reduce this matrix. This has already been proven before. We applied the four aforementioned matrix reduction algorithms, including BESTA, to this matrix without any stopping condition until we obtained the smallest possible matrix of dimensions 3x3. At each reduction step, the matrix is reduced by one state and the parameters of the reduced matrix queue are displayed.

Figure 10 shows the normalized evaluation of the four metrics for each reduction step using the four reduction algorithms. We can quickly see that the BESTA algorithm intersects give the best results quite for all reduction steps (stars corresponding to the BESTA algorithm are near to the solid line that corresponds to the original value of the metrics). Therefore, we can conclude that if the stopping condition is based on a combination of the metrics (Average Service Time, Service Rate, and Average Waiting Time), we can assert that the BESTA algorithm may stop between the 7th and the 10th step to give the an optimal matrix (which is capable of representing the M/M/1 Markov chain in terms of the four metrics) between 13x13 and 10x10. In fact, for these

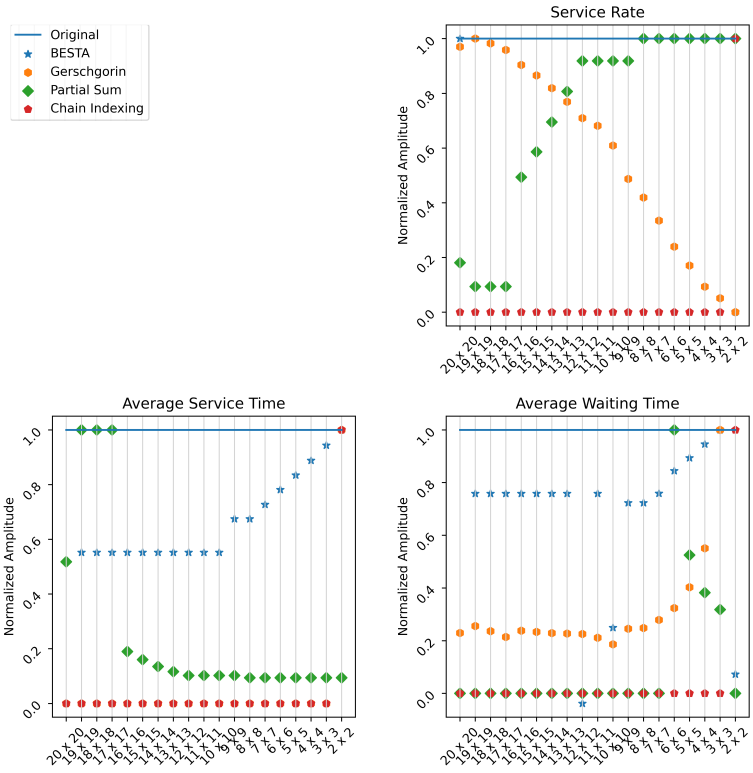

Fig. 10: Comparison of the BESTA algorithm against the three following popular Markov chain reduction algorithms: Gerschgorin, Chain Indexing and partial Sum algorithms. The curve have been normalized and the Original ones allows to display the metrics (Average Waiting Time, Average Service Time and Service Rate) of the original 20x20 M/M/1 chain (without reduction).

dimensions, the metrics are closest to the original ones if the BESTA method is chosen. We can notice out that in the of the service rate, the BESTA and the partial sum give the same values.

4.3.2 The Stopping Condition Issue

The stopping condition ϵ plays an important role in balancing the trade-off and must keep track of the process between different iterations in order to be able to plan the end of iterations when the difference between the current computed KL and the previous one is large enough to decide to stop the loop.

We introduce this stopping condition, which consists in finding the appropriate ϵ which can be defined as a tracking error that may not exceed a specific value (expressed as a percentage, for example). This stopping condition allows the algorithm progress to be assessed in a relative fashion by comparing the result of different iterations. Typically, the stopping condition is invoked at the end of an iteration of the algorithm. At this point, it will be decided whether to continue the execution of the algorithm or abort it. In addition to detecting situations where algorithms should be stopped, stopping criteria should be lightweight in terms of computational complexity. The simplest approach to stopping is to calculate how well the state of the current algorithm satisfies a given quality threshold. Local criteria use information that is not present in the context of the iteration but only in the context of this local criterion. In our case, the stopping condition, which is based on ϵ is expressed by the difference between the current KL and the previous one. Empirically, we have defined that to obtain the optimal aggregated Markov chain, the previous deviation of the KL should not exceed 50%.

5 Related Work

There are several effective algorithms to reduce a Markov chain, including the following non-exhaustive list: (i) The GTH reduction algorithm [42]: This algorithm uses the power-balancing method to reduce a Markov chain by eliminating transient states (ii) The PageRank reduction algorithm [38]: This algorithm uses a version of the famous Google algorithm to reduce a Markov chain using the notion of visit probability to eliminate transient states (iii) The Gerschgorin reduction algorithm [24]: This algorithm uses the Gerschgorin theorem to reduce a Markov chain by identifying strongly connected states and combining them into a single state (iv) The lumping reduction algorithm [39]: This algorithm uses a state grouping method to reduce a Markov chain by combining similar states into a single state (v) The Kronecker reduction algorithm [26]: This algorithm uses the Kronecker decomposition to reduce a Markov chain by identifying recurrent sub-chains and grouping them into a single state (vi) The QBD reduction algorithm [17]: This algorithm uses the quasi-banded diagonal block method to reduce a Markov chain to a quasi-diagonal form that can be more easily solved (vii) The Partial Sum (used in [5]) that involves summing a finite number of terms of a power series to obtain an approximation of the reduced transition probability matrix. and (viii) Markov Chain Indexing Method [23] that groups together states that have similar properties or behavior. Whereas the mentioned algorithms operate by either eliminating transient states, identifying strongly connected states, or grouping similar states based on various criteria, the proposed approach in this paper uses a KL-based reduction prioritizes preserving the probabilistic structure of the Markov chain by minimizing the information loss between the original and reduced representations. This approach ensures that the reduced

Markov matrix maintains fidelity to the original while achieving dimensionality reduction.

As mentioned in the Introduction section, the lumpability process can be based on the KL divergence metric (or an equivalent metric) as can be found in [19,13,18,12,15,11,36,37,31,30,29]. We may highlight that the last three articles [31,30,29] mainly address the aggregation of Markov chains that appear in Markov decision process-based reinforcement learning as in [20]. In [12], due to the large number of partition functions, the partition problem is addressed by combining a parameterization of the randomized partition policy and a simulation-based gradient descent algorithm instead of using Rank and Selection (R&S) or other classic optimization approaches [34]. In the parameterization approach, a vector $\theta = [\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N] \in \Re^N$ is introduced [25] and associated with a randomized partition policy $\nu_\phi(i, \theta)$ where $\sum_\phi \nu_\phi(i, \theta) = 1$.

Recent advancements in dimensionality reduction techniques for both continuous and discrete-time Markov chains have introduced the utilization of the graph transformation algorithm for computing the matrix of microstate First Transition Probabilities (MFTPs) within the original Markov chain [21]. The authors illustrate that the graph transformation, typically applied in continuous-time Markov chains, can be extended to discrete-time counterparts, facilitating model representation and numerical analysis. Despite the demonstration of this approach's viability on a 32-dimensional matrix, the findings exhibit a noteworthy degree of resilience. The proposed study shares similarities with this research; however, it aims to validate its methodology on larger matrices.

The paper aims to approximate the original Markov process, retaining its key characteristics while simplifying its complexity. Through a user-centric heuristic, it proposes state aggregation to decrease dimensionality while preserving the essence of the original chain. Utilizing total variation metrics, the approach seeks to derive a lower-dimensional process that closely mimics the behavior of the original Markov process, thereby reducing computational complexity.

6 Conclusion

The contribution of this paper is to provide a method for reducing Markov chains based on a reduction criterion that can be either the desired final size of the chain or a percentage of similarity. We present a concrete example to demonstrate that this new method allows for reducing a Markov chain while preserving the semantics of the system modeled by the chain. This paper proposes a novel approach to speeding up the process of reducing large ergodic Markov chains by combining deterministic and heuristic algorithms. The approach involves balancing two conflicting objectives: (i) minimizing the size of the computed aggregation, and (ii) ensuring that the aggregation is as similar as possible to the original chain to preserve its global dynamical properties. The paper utilizes several conceptual characteristics, including (i) the KL met-

ric and the lumpability concept for comparing Markov chains, (ii) the notion of mean first passage time associated with ergodic Markov chains, and (iii) a property introduced in [22] that links the lumpability concept with the mean first passage time notion. The proposed algorithms have been implemented in Python and compared using a benchmark consisting of a set of large Markov chains, which are available in the accompanying Git repository [8].

The comparison was performed through a set of experiments that confirmed the following: (i) the benefits of deterministic improvement, as the best partitioning was obtained when the class number was equal to $k = n - 1$, where n is the number of states in the Markov chain; (ii) the efficiency of a heuristic criterion and the cost-effective applicability of the reduction Markov chain algorithm, even on large chains; and (iii) the originality of the solution for resolving the conflicting problem, achieved by controlling the aggregation algorithm using a threshold (ϵ) or a target number of aggregated states.

Future work will concentrate on several investigations, briefly described as follows: (i) We will seek to compare the proposed approach to reduce a given Markov chain by attempting to guide a simulation-based method towards relevant areas of the state space when searching for the optimal partition (complemented by an approach based on Neural Nets or Support Vector Machines to minimize the risk of not exploring relevant parts of the state space); (ii) We plan to consider semi-Markov models [2], 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 [40] formalism's properties fit very well with the resolution of temporal Markov problems [9], and (iii) We envision using metaheuristic-based chaotic structures [16]. Chaos theory is a novel approach that has been used in various applications, such as multi-criteria optimization problems [35]. Although the method has some limitations and there are cases where it is unsuccessful, several analytical and experimental results obtained by us classify this new method as a potentially useful tool in applications.

Conflict of Interest

The authors declare that they have no conflict of interest.

Data Availability Statement

The data used to support the findings of this study are available at `https://github.com/capocchi/MarkovLumping`.

References

1. Understanding and Choosing the Right Probability Distributions, pp. 899–917. John Wiley & Sons, Ltd (2012). DOI https://doi.org/10.1002/9781119197096.app03

2. Akimenko, T.A., Larkin, E.V.: The method of successive simplifications of the semi-markov process. In: 2019 8th Mediterranean Conference on Embedded Computing (MECO), pp. 1–5 (2019). DOI 10.1109/MECO.2019.8760165
3. Amjad, R.A., Blöchl, C., Geiger, B.C.: A generalized framework for kullback–leibler markov aggregation. IEEE Transactions on Automatic Control **65**(7), 3068–3075 (2020). DOI 10.1109/TAC.2019.2945891
4. Aoki, M.: Some approximation methods for estimation and control of large scale systems. IEEE Transactions on Automatic Control **23**(2), 173–182 (1978). DOI 10.1109/TAC.1978.1101705
5. Barsotti, F., De Castro, Y., Espinasse, T., Rochet, P.: Estimating the transition matrix of a markov chain observed at random times. Statistics & Probability Letters **94**, 98–105 (2014). DOI https://doi.org/10.1016/j.spl.2014.07.009
6. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. Wiley-Interscience, USA (1998)
7. Buchholz, P.: Exact and ordinary lumpability in finite markov chains. Journal of Applied Probability **31**(1), 59–75 (1994)
8. Capocchi, L.: Git repo to reproduce results. `https://github.com/capocchi/MarkovLumping`. Online; accessed september 18 2021
9. Cardelli, L., Grosu, R., Larsen, K.G., Tribastone, M., Tschaikowski, M., Vandin, A.: Algorithmic minimization of uncertain continuous-time markov chains. IEEE Transactions on Automatic Control **68**(11), 6557–6572 (2023). DOI 10.1109/TAC.2023.3244093
10. Cho, G., Meyer, C.: Markov chain sensitivity measured by mean first passage times. Linear Algebra and its Applications **316**(1-3), 21–28 (2000). DOI 10.1016/S0024-3795(99)00263-3. URL `http://meyer.math.ncsu.edu/Meyer/PS_Files/SensitivityByMFP.pdf`
11. Deng, K., Huang, D.: Model reduction of markov chains via low-rank approximation. In: 2012 American Control Conference, pp. 2651–2656 (2012). DOI 10.1109/ACC.2012.6314781
12. Deng, K., Mehta, P.G., Meyn, S.P.: A simulation-based method for aggregating markov chains. In: Proc. of the 48th IEEE Conference on Decision and Control, combined withe the 28th Chinese Control Conference, December 16-18, Shanghai, China, pp. 4710–4716. IEEE (2009). DOI 10.1109/CDC.2009.5400533
13. Deng, K., Mehta, P.G., Meyn, S.P.: Optimal kullback leibler aggregation via spectral theory of markov chains. IEEE Transactions on Automatic Control **56**(12), 2793–2808 (2011). DOI 10.1109/TAC.2011.2141350
14. Deng, K., Mehta, P.G., Meyn, S.P., Vidyasagar, M.: A recursive learning algorithm for model reduction of hidden markov models. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference, pp. 4674–4679 (2011). DOI 10.1109/CDC.2011.6160826
15. Deng, K., Sun, Y., Mehta, P.G., Meyn, S.P.: An information-theoretic framework to aggregate a markov chain. In: American Control Conference, St. Louis, Missouri, USA, June 10-12, pp. 731–736. IEEE (2009). DOI 10.1109/ACC.2009.5160607
16. Dhawale, D., Kamboj, V., Anand, P.: An effective solution to numerical and multi-disciplinary design optimization problems using chaotic slime mold algorithm. Engineering with Computers pp. 1–39 (2021). DOI 10.1007/s00366-021-01409-4
17. Ellaia, R., Imine, A., Sorel, Y.: An efficient quasi-birth-and-death reduction algorithm for markovian models. Performance Evaluation **153**, 102199 (2021). DOI 10.1016/j.peva.2021.102199
18. Geiger, B.C., Petrov, T., Kubin, G., Koeppl, H.: Optimal kullback-leibler aggregation via information bottleneck. IEEE Transactions on Automatic Control **60**(4), 1010–1022 (2015). DOI 10.1109/TAC.2014.2364971
19. Geiger, B.C., Wu, Y.: Higher-order kullback-leibler aggregation of markov chains. In: SCC 2017; 11th International ITG Conference on Systems, Communications and Coding, pp. 1–6 (2017)
20. Jia, Q.: On state aggregation to approximate complex value functions in large-scale markov decision processes. IEEE Transactions on Automatic Control **56**(2), 333–344 (2011). DOI 10.1109/TAC.2010.2052697

21. Kannan, D., Sharpe, D.J., Swinburne, T.D., Wales, D.J.: Optimal dimensionality reduction of markov chains using graph transformation. The Journal of Chemical Physics **153**(24), 244108 (2020). DOI 10.1063/5.0025174
22. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Springer (1976)
23. Kumar, K.N., Reddy, K.K.: Markovian semantic indexing:application to online image retrieval system (2014)
24. Mall, R., Mehrkanoon, S., Suykens, J.A.: Identifying intervals for hierarchical clustering using the gershgorin circle theorem. Pattern Recognition Letters **55**, 1–7 (2015). DOI https://doi.org/10.1016/j.patrec.2014.12.007
25. Marbach, P., Tsitsiklis, J.N.: Simulation-based optimization of markov reward processes. In: Proc. of the 37th IEEE Conference on Decision and Control, vol. 3, pp. 2698–2703 vol.3 (1998). DOI 10.1109/CDC.1998.757861
26. Meyer, C.D., Balaji, S.: Efficient computation of stationary probabilities for very large markov chains. SIAM Journal on Scientific Computing **42**(1), B107–B132 (2020). DOI 10.1137/19M1282698
27. Pillai, S., Suel, T., Cha, S.: The perron-frobenius theorem: some of its applications. IEEE Signal Processing Magazine **22**(2), 62–75 (2005). DOI 10.1109/MSP.2005.1406483
28. Rached, Z., Alajaji, F., Campbell, L.L.: The kullback-leibler divergence rate between markov sources. IEEE Transactions on Information Theory **50**(5), 917–921 (2004). DOI 10.1109/TIT.2004.826687
29. Sledge, I., Príncipe, J.: Reduction of markov chains using a value-of-information-based approach. Entropy **21**(4), 349 (2019). DOI 10.3390/e21040349
30. Sledge, I.J., Emigh, M.S., Príncipe, J.C.: Guided policy exploration for markov decision processes using an uncertainty-based value-of-information criterion. IEEE Transactions on Neural Networks and Learning Systems **29**(6), 2080–2098 (2018). DOI 10.1109/TNNLS.2018.2812709
31. Sledge, I.J., Príncipe, J.C.: Analysis of agent expertise in ms. pac-man using value-of-information-based policies. IEEE Transactions on Games **11**(2), 142–158 (2019). DOI 10.1109/TG.2018.2808201
32. Sledge, I.J., Príncipe, J.C.: Reduction of markov chains using a value-of-information-based approach. Entropy **21**(4) (2019). DOI 10.3390/e21040349
33. Sumita, U., Rieders, M.: First passage times and lumpability of semi-markov processes. Journal of Applied Probability **25**, 675–687 (1988). DOI 10.1017/S0021900200041462
34. Swisher, J., Hyden, P., Jacobson, S., Schruben, L.: A survey of simulation optimization techniques and procedures. In: Proc. of the Simulation Conference, Winter, vol. 1, pp. 119 –128 vol.1 (2000)
35. Tang, R., Fong, S., Dey, N.: Metaheuristics and Chaos Theory (2018). DOI 10.5772/intechopen.72103
36. Vidyasagar, M.: Reduced-order modeling of markov and hidden markov processes via aggregation. In: 49th IEEE Conference on Decision and Control, pp. 1810–1815 (2010). DOI 10.1109/CDC.2010.5717206
37. Vidyasagar, M.: A metric between probability distributions on finite sets of different cardinalities and applications to order reduction. IEEE Transactions on Automatic Control **57**(10), 2464–2477 (2012). DOI 10.1109/TAC.2012.2188423
38. Wu, C., Cai, Y., Zhang, H.: Efficient markov chain reduction using the pagerank algorithm. Journal of Computational Science **26**, 86–93 (2018). DOI 10.1016/j.jocs.2018.09.009
39. Yin, J., Chow, S.N., Petzold, L.R.: Lumping analysis in the dimensionality reduction of markov state models. Multiscale Modeling & Simulation **16**(1), 392–417 (2018). DOI 10.1137/17M1135069
40. Zeigler, B., Muzy, A., Kofman, E.: Theory of Modeling and Simulation. 3rd Edition. Academic Press, Inc., Orlando, FL, USA (2018)
41. Zhang, A., Wang, M.: Spectral state compression of markov processes. IEEE Transactions on Information Theory **66**(5), 3202–3231 (2020). DOI 10.1109/TIT.2019.2956737
42. Zhao, Y.Q.: Gth algorithm, censored markov chains, and rg-factorization (2021)