

Formal error bounds for the state space reduction of Markov chains

Fabian Michel ^{ID}*, Markus Siegle ^{ID}

Universität der Bundeswehr München, Institut für Technische Informatik, Werner-Heisenberg-Weg 39, Neubiberg, 85579, Germany

ARTICLE INFO

Keywords:

Markov chains
State space reduction
Formal error bounds
Aggregation
Lumpability

ABSTRACT

We study the approximation of a Markov chain on a reduced state space, for both discrete- and continuous-time Markov chains. In this context, we extend the existing theory of formal error bounds for the approximated transient distributions. In the discrete-time setting, we bound the stepwise increment of the error, and in the continuous-time setting, we bound the rate at which the error grows. In addition, the same error bounds can also be applied to bound how far an approximated stationary distribution is from stationarity. As a special case, we consider aggregated (or lumped) Markov chains, where the state space reduction is achieved by partitioning the state space into macro states. Subsequently, we compare the error bounds with relevant concepts from the literature, such as exact and ordinary lumpability, as well as deflatability and aggregatability. These concepts provide stricter than necessary conditions for settings in which the aggregation error is zero. We also present possible algorithms for finding suitable aggregations for which the formal error bounds are low, and we analyze first experiments with these algorithms on a range of different models.

1. Introduction

State aggregation in dynamic systems has been studied extensively since the 1960s (see [1]). Due to the curse of dimensionality, models with large state spaces are often computationally intractable without state space reduction. One basic reduction technique is to aggregate multiple states into a single state in the aggregated model, and conditions under which an aggregated Markov chain is again a Markov chain are well known (see strong and weak lumpability in [2,3]). Various cases where exact transient or stationary probabilities of the original model can be derived from an aggregated model have been identified and analyzed (see, e.g. [4]).

However, formal error bounds for the approximation error when exact aggregation is not possible have only been studied rarely. [4] already gave upper and lower bounds for the transient distribution of a Markov chain which are derived from an aggregated model. More than 25 years later, [5] has presented improved bounds for the transient distribution of discrete-time Markov chains, which can also be applied to continuous-time Markov chains via uniformization.

Our contribution. Our main contribution is the development of a more general theory for calculating error bounds in a wider set of contexts. [5] limited its analysis to aggregations where the state space is partitioned into groups of states forming a macro state in the aggregated model, with every state in a group weighted equally. The theory in [5] was only fully developed for discrete time. We will consider a much more abstract way of state space reduction where no intuitive meaning for an “aggregated state” exists — every original state can be a part of every aggregated state, where the influence on an aggregated state is given by some (positive or negative) weight. In addition, most of our results hold in discrete and in continuous time. It is therefore possible to combine our error bounds with most aggregation approaches, including previously uncovered cases such as fuzzy clustering in [6] where each state may belong to multiple different aggregated states and where the membership in the different aggregates is represented by

* Corresponding author.

E-mail address: fabian.michel@unibw.de (F. Michel).

<https://doi.org/10.1016/j.peva.2024.102464>

Received 14 August 2024; Received in revised form 19 November 2024; Accepted 11 December 2024

Available online 18 December 2024

0166-5316/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

non-negative weights summing to one. Moreover, the theoretical insights can also be the basis for future generalizations, e.g. to Markov processes with continuous state spaces.

Our second contribution is the presentation of new algorithms which try to identify settings in which the error bounds are low. The algorithms are based on a general condition allowing lossless aggregation which had previously not been considered in the context of aggregation, and which we discovered with the help of the newly developed theory. Due to computational tractability considerations, we were not yet able to exploit the full flexibility in the aggregation schemes compatible with our bounds. Instead, the presented algorithms only look for a specific subset of the cases allowing aggregations with low error bounds. One algorithm is an extension of [7] with improved performance and numerical stability (and for which the previously known error bounds were not applicable), and one is based on theory which was already developed in [8]. Our experiments show that the error bounds can be useful in practical applications, even without employing additional techniques such as adaptive reaggregation and truncation as used in [5]. Research on the combination with these techniques is left for future work.

The present paper is an extension of our conference paper [9], where the most important concepts behind the error bounds were introduced in a more condensed manner. This paper extends [9] to a more general setting for aggregation, includes some remarks on stationary error bounds, covers some of the theoretical properties in more detail, and gives full pseudo-code and justifications for both of the presented algorithms. In addition, we now provide a more extensive experimental evaluation.

Paper organization. We extend the theory developed in [5] to support a more general way of disaggregation in Section 3.1 and to the continuous-time domain without falling back on uniformization in Section 3.2. Subsequently, in Section 3.3, we analyze the cases where the error bounds for the transient distribution are zero. We also consider how the bounds relate to approximations of the stationary distribution in Section 3.4. Furthermore, we show that the error bounds from [5] are tight for transient distributions in general in Section 3.5, but not for stationary distributions. To complete the theoretical analysis, we then compare the error bounds with lumpability concepts from [2,4,7,10] in Section 4. Many of these types of lumpability imply, but are not equivalent to the error bound being zero.

In Section 5, we present two different algorithms, one an extension of [7] and one based on [8], with the goal to identify an aggregation resulting in low error bounds. We then apply these algorithms to examples, some from the literature and some randomly generated, and we analyze the results in Section 6.

2. Preliminaries

2.1. Linear algebra

Let $v \in \mathbb{R}^m$ be a vector and $A \in \mathbb{R}^{m \times n}$ a matrix. All considered vectors are column vectors. We use \cdot^T to denote the transpose of a matrix or vector, so for the column vector v , the transpose v^T is a row vector. $|v|$ and $|A|$ denote the vector and matrix where the absolute value was applied component-wise. For vectors $u, v \in \mathbb{R}^m$, we use $\langle u, v \rangle := \sum_{i=1}^m u(i)v(i) = u^T v$ to denote the usual inner product of two vectors, where $v(i)$ is the i th entry of v . We further use $\mathbf{1}_m$ to denote the column vector $(1, \dots, 1)^T \in \mathbb{R}^m$ and $\mathbf{1}_{m \times n} \in \mathbb{R}^{m \times n}$ to denote a matrix where every entry is one. In contrast, $I_n \in \mathbb{R}^{n \times n}$ denotes the identity matrix. We set $\|v\|_1 := \langle |v|, \mathbf{1}_m \rangle = \sum_{i=1}^m |v(i)|$ and $\|A\|_\infty = \max_{i=1, \dots, m} \sum_{j=1}^n |A(i, j)|$. We will later use that the matrix norm $\|\cdot\|_\infty$ is submultiplicative: for matrices A and B , we have $\|AB\|_\infty \leq \|A\|_\infty \cdot \|B\|_\infty$ (see [11, equations (2.3.3) and (2.3.10)]). Note that $\|v^T\|_\infty = \|v\|_1$ if we interpret v as a matrix. However, to avoid using different notation for the norms of row and column vectors, we will only use $\|\cdot\|_1$ to denote the norm of a (row or column) vector, i.e. $\|v\|_1 = \|v^T\|_1 = \sum_{i=1}^m |v(i)|$. As $\|\cdot\|_1$ is only used for vectors and $\|\cdot\|_\infty$ only for matrices in this paper, this will (hopefully) not cause any confusion.

Lemma 1. For a vector $v \in \mathbb{R}^m$ and a matrix $A \in \mathbb{R}^{m \times n}$, we have:

$$\|v^T A\|_1 \leq \begin{cases} \langle |v|, |A| \cdot \mathbf{1}_n \rangle \\ \|v\|_1 \cdot \|A\|_\infty \end{cases}$$

Proof. We have

$$\begin{aligned} \|v^T A\|_1 &= \sum_{j=1}^n \left| \sum_{i=1}^m v(i) \cdot A(i, j) \right| \leq \sum_{j=1}^n \sum_{i=1}^m |v(i)| \cdot |A(i, j)| \\ &= \sum_{i=1}^m |v(i)| \sum_{j=1}^n |A(i, j)| \begin{cases} \langle |v|, |A| \cdot \mathbf{1}_n \rangle \\ \leq \left(\sum_{i=1}^m |v(i)| \right) \left(\max_{i=1, \dots, m} \sum_{j=1}^n |A(i, j)| \right) = \|v\|_1 \cdot \|A\|_\infty \end{cases} \quad \square \end{aligned}$$

We will use the term stochastic matrix to denote matrices $A \in \mathbb{R}^{m \times n}$ (where it is not necessarily the case that $m = n$) whose entries are all non-negative and where every row sums to 1 (i.e. $A \cdot \mathbf{1}_n = \mathbf{1}_m$).

Lemma 2. Let $A \in \mathbb{R}^{n \times n}$ and $t \geq 0$. Then

$$\|e^{At}\|_\infty \leq e^{t\|A\|_\infty}$$

Proof. By submultiplicativity of $\|\cdot\|_\infty$ for matrices, we have

$$\|e^{At}\|_\infty = \left\| \sum_{k \geq 0} \frac{t^k A^k}{k!} \right\|_\infty \leq \sum_{k \geq 0} \left\| \frac{t^k A^k}{k!} \right\|_\infty = \sum_{k \geq 0} \frac{t^k}{k!} \|A\|_\infty^k \leq \sum_{k \geq 0} \frac{t^k}{k!} \|A\|_\infty^k = e^{t\|A\|_\infty} \quad \square$$

2.2. Markov chains and state space reduction

We consider time-homogeneous discrete- and continuous-time Markov chains (DTMCs and CTMCs) on the finite state space $S = \{1, \dots, n\}$. The dynamics are given by the stochastic transition matrix $P \in \mathbb{R}^{n \times n}$ for DTMCs, where we have $P(i, j) = \mathbb{P}[X_{k+1} = j | X_k = i]$ if X_k denotes the state of the DTMC at time k . For CTMCs, the dynamics are defined via the generator matrix $Q \in \mathbb{R}^{n \times n}$, where $Q(i, j)$ is the transition rate from i to j , and $Q(i, i) = -\sum_{j \neq i} Q(i, j)$. Given an initial distribution $p_0 \in \mathbb{R}^n$, the transient distribution of a DTMC (respectively CTMC) is given by $p_k^\top = p_0^\top P^k$ (respectively $p_t^\top = p_0^\top e^{Qt}$).

We want to reduce the state space of the Markov chain to speed up computation of various properties. We often refer to state space reduction as aggregation, even though we take a very abstract view of aggregation: there are not necessarily groups of states which are aggregated into one macro state. Instead, we define the aggregation of a Markov chain with an aggregated state space of dimension m (where $m \leq n$) as follows: given a disaggregation matrix $A \in \mathbb{R}^{m \times n}$ (which can be arbitrary), an (arbitrary) aggregated step matrix $\Pi \in \mathbb{R}^{m \times m}$ for DTMCs, an (arbitrary) aggregated evolution matrix $\Theta \in \mathbb{R}^{m \times m}$ for CTMCs, and an (arbitrary) aggregated initial vector $\pi_0 \in \mathbb{R}^m$, we approximate the dynamics of the original chain by setting $\tilde{p}_k^\top := \pi_k^\top A := \pi_0^\top \Pi^k A$ and $\tilde{p}_t^\top := \pi_t^\top A := \pi_0^\top e^{\Theta t} A$. \tilde{p}_k and \tilde{p}_t are intended to approximate the transient distributions of the original Markov chains, i.e. p_k and p_t .

In this abstract view of aggregation, we do not require that Π is stochastic, that Θ is a generator or that π_0 is a probability distribution. However, intuitively, Π should be an approximation of the step dynamics of the DTMC in a lower-dimensional state space, and it often makes sense to choose a stochastic matrix Π , which would imply that we approximate the original chain with a DTMC with fewer states. The more abstract view with arbitrary matrices Π allows us, in theory, to approximate the step dynamics in the lower-dimensional state space with any linear map. We opt for the greater generality since most of the results presented in this paper already hold in this very abstract setting.

In practical applications, a typical choice would be to choose a stochastic matrix Π , a generator matrix Θ , a probability distribution π_0 , and a stochastic matrix A (the latter guarantees that \tilde{p}_k is a probability distribution as well). We call A disaggregation matrix since A describes how to blow up the aggregated transient distribution π_k to the full-state-space approximation \tilde{p}_k via the equation $\tilde{p}_k^\top = \pi_k^\top A$, which corresponds to disaggregating π_k .

The most commonly studied type of aggregation is even more restrictive in the possible choices for Π , Θ , A and π_0 . In most published approaches, the state space S of the original chain is partitioned into aggregates by some partition $\Omega = \{\Omega_1, \dots, \Omega_m\}$ of S , with $\sigma \in \Omega$ being a subset of S which represents all states belonging to one aggregate. The aggregation function $\omega : S \rightarrow \Omega$ maps a state s to the aggregate to which s belongs, i.e. $s \in \omega(s)$. Instead of an arbitrary disaggregation matrix A , one defines probability distributions $\alpha_\sigma \in \mathbb{R}^n$ with support on $\sigma \in \Omega$. As a shorthand, we write $\alpha(s) := \alpha_{\omega(s)}(s)$. The value $\alpha(s)$ should approximate the conditional probability of being in state s when the chain is in the aggregate $\omega(s)$, i.e. the probability $\mathbb{P}[X_k = s | X_k \in \omega(s)]$. This probability is in general dependent on time, but commonly, only time-independent approximations α are considered. α_σ can be thought of as a probability distribution which splits the probability mass of the aggregate σ among its constituting states in the disaggregation phase, and can in general be chosen by the user. We give some heuristics on how to choose the α distributions later, e.g. in (15). One can then define the disaggregation matrix A and the aggregation matrix Λ as follows:

$$A = \begin{pmatrix} | & & | \\ \mathbb{1}_{\Omega_1} & \dots & \mathbb{1}_{\Omega_m} \\ | & & | \end{pmatrix} \in \mathbb{R}^{n \times m}, \quad \Lambda = \begin{pmatrix} -\alpha_{\Omega_1}^\top & - \\ \vdots & \\ -\alpha_{\Omega_m}^\top & - \end{pmatrix} \in \mathbb{R}^{m \times n} \quad (\text{note: } \Lambda A = I)$$

where $\mathbb{1}_\sigma \in \mathbb{R}^n$ is defined by

$$\mathbb{1}_{\sigma}(s) = \begin{cases} 1 & \text{if } s \in \sigma \\ 0 & \text{otherwise} \end{cases}$$

A natural definition for Π and Θ is then given by $\Pi = \Lambda A$ and $\Theta = \Lambda Q \Lambda$, which will ensure that Π is stochastic and that Θ is a generator. In this case, $\Pi(\rho, \sigma)$ for $\rho, \sigma \in \Omega$ is an approximation of the probability to transition from one aggregate state into another, that is, an approximation of $\mathbb{P}[X_{k+1} \in \sigma | X_k \in \rho]$. Note that this probability may also depend on time (i.e. on k) in general, in contrast to the probability $\mathbb{P}[X_{k+1} = s | X_k = r]$ for $r, s \in S$. However, we again consider only time-independent approximations of $\mathbb{P}[X_{k+1} \in \sigma | X_k \in \rho]$. Similarly, for CTMCs, we should have

$$\Theta(\rho, \sigma) \approx \lim_{u \rightarrow 0} \frac{\mathbb{P}[X_{t+u} \in \sigma | X_t \in \rho]}{u}$$

if we aim at a faithful approximation of the dynamics. Furthermore, $\pi_0^\top = p_0^\top \Lambda$ is the natural choice for the initial distribution when working with actual aggregates.

As an example, consider the DTMC on the state space $\{1, 2, 3\}$ given by

$$P = \frac{1}{4} \cdot \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix}, \quad p_0 = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)^\top$$

A possible choice of aggregation using the above method would be

$$A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \Lambda = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Pi = A P \Lambda = \frac{1}{8} \cdot \begin{pmatrix} 5 & 3 \\ 6 & 2 \end{pmatrix}, \quad \pi_0 = \Lambda^\top p_0 = \left(\frac{3}{4}, \frac{1}{4} \right)^\top$$

corresponding to the aggregates $\Omega = \{\{1, 2\}, \{3\}\}$ and to $\alpha(1) = \alpha(2) = \frac{1}{2}, \alpha(3) = 1$. We obtain the following transient distribution at time 1 and its approximation:

$$p_1^\top = p_0^\top P = \left(\frac{5}{16}, \frac{3}{8}, \frac{5}{16} \right), \quad \tilde{p}_1^\top = \pi_0^\top \Pi A = \left(\frac{21}{64}, \frac{21}{64}, \frac{11}{32} \right)$$

Definition 3. Given a partition Ω of the state space of a DTMC or CTMC, we call a probability distribution p on the state space S **compatible** with distributions α_σ with support on $\sigma \in \Omega$ if $p^\top \Lambda A = p^\top$.

Compatibility of p and the distributions α means that

$$\alpha(s) = \frac{p(s)}{\sum_{s' \in \omega(s)} p(s')} \text{ for all } s \in S \text{ s.t. } \sum_{s' \in \omega(s)} p(s') > 0$$

To make it easier to identify this special case of state space reduction via the partition Ω , we will always use r, s to denote the states of the original chain and ρ, σ to denote the aggregates in this context, and we will call this type of aggregation **weighted state space partitioning**. In the more abstract context where we allow arbitrary Π, Θ, A and π_0 , we will use i, j to denote the states of the original chain and χ, ϕ to denote the abstract states of the aggregated chain.

3. Bounding the approximation error

3.1. Error bounds for DTMCs

In this section, we derive formal error bounds for the difference between the actual transient distribution p_k of the Markov chain and the approximation \tilde{p}_k of this distribution, obtained after state space reduction. We loosely follow [5] in our derivation, but a few adaptations are necessary as we consider a more general setting for aggregation. [4, Theorem 11] also gave error bounds similar to [5], but the bounds in [5] are slightly better. Throughout Section 3.1, we consider the disaggregation matrix A , the aggregated step matrix Π , and the aggregated initial vector π_0 as arbitrary but fixed.

Theorem 4. Let e_k be the error vector after k steps, i.e.

$$e_k^\top = \tilde{p}_k^\top - p_k^\top = \pi_0^\top \Pi^k A - p_0^\top P^k$$

Then, the following hold:

- (i) $\|e_k\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + \sum_{j=0}^{k-1} \langle |\pi_j|, |\Pi A - A P| \cdot \mathbf{1}_n \rangle$
- (ii) $\|e_k\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + \|\pi_0\|_1 \cdot \|\Pi A - A P\|_\infty \cdot \begin{cases} \frac{\|\Pi\|_\infty^k - 1}{\|\Pi\|_\infty - 1} & \text{if } \|\Pi\|_\infty \neq 1 \\ k & \text{otherwise} \end{cases}$
- (iii) if Π is stochastic and π_0 is a probability distribution, then $\|e_k\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + k \cdot \|\Pi A - A P\|_\infty$

Proof. We want to give a bound on $\|e_k\|_1$, where $e_k = \tilde{p}_k - p_k$ is the error vector after k steps. We have

$$\begin{aligned} \|e_k\|_1 &= \|\pi_0^\top \Pi^k A - p_0^\top P^k\|_1 = \left\| \underbrace{(\pi_0^\top \Pi^{k-1} A - p_0^\top P^{k-1})}_{e_{k-1}^\top} \cdot P + \underbrace{\pi_0^\top \Pi^{k-1}}_{\pi_{k-1}^\top} (\Pi A - A P) \right\|_1 \\ &\leq \|e_{k-1}^\top \cdot P\|_1 + \|\pi_{k-1}^\top \cdot (\Pi A - A P)\|_1 \stackrel{\text{Lemma 1}}{\leq} \|e_{k-1}\|_1 \cdot \underbrace{\|P\|_\infty}_{=1} + \|\pi_{k-1}^\top \cdot (\Pi A - A P)\|_1 \\ &\stackrel{\text{Lemma 1}}{\leq} \begin{cases} \|e_{k-1}\|_1 + \langle |\pi_{k-1}|, |\Pi A - A P| \cdot \mathbf{1}_n \rangle \\ \|e_{k-1}\|_1 + \|\pi_{k-1}\|_1 \cdot \|\Pi A - A P\|_\infty \end{cases} \end{aligned} \tag{1}$$

We can continue with the first bound as follows:

$$\|e_k\|_1 \leq \|e_0\|_1 + \sum_{j=0}^{k-1} \langle |\pi_j|, |\Pi A - A P| \cdot \mathbf{1}_n \rangle$$

which proves (i). Using that

$$\|\pi_{k-1}\|_1 = \|\pi_0^\top \cdot \Pi^{k-1}\|_1 \stackrel{\text{Lemma 1 \& submultiplicativity}}{\leq} \|\pi_0\|_1 \cdot \|\Pi\|_\infty^{k-1}$$

we can also continue with the second bound in (1), obtaining

$$\begin{aligned} \|e_k\|_1 &\leq \|e_0\|_1 + \sum_{j=0}^{k-1} \|\pi_0\|_1 \cdot \|\Pi\|_\infty^j \cdot \|\Pi A - AP\|_\infty \\ &= \|\pi_0^\top A - p_0^\top\|_1 + \|\pi_0\|_1 \cdot \|\Pi A - AP\|_\infty \cdot \sum_{j=0}^{k-1} \|\Pi\|_\infty^j \\ &= \|\pi_0^\top A - p_0^\top\|_1 + \|\pi_0\|_1 \cdot \|\Pi A - AP\|_\infty \cdot \begin{cases} \frac{\|\Pi\|_\infty^k - 1}{\|\Pi\|_\infty - 1} & \text{if } \|\Pi\|_\infty \neq 1 \\ k & \text{otherwise} \end{cases} \end{aligned}$$

which proves (ii) and (iii). \square

Some remarks are in order: in [5], the bounds look quite different on first sight (compare with [5, equations (17) and (18)]). However, they are actually a special case of the above bounds. In fact, we can define $\tau(\chi)$ for $\chi \in \{1, \dots, m\}$ in analogy to [5] as follows: $\tau(\chi) := (|\Pi A - AP| \cdot \mathbf{1}_n)(\chi)$ (note that $|\Pi A - AP| \cdot \mathbf{1}_n$ is a column vector with m entries). We can interpret $\tau(\chi)$ as the error caused by “aggregate” χ , even though the word “aggregate” does not have an intuitive meaning in our general setting for aggregations (there are no such things as aggregates consisting of a certain group of states in our case). In [5], the considered aggregations correspond almost exactly to what was defined as weighted state space partitioning in Section 2.2, with the following exceptions: the α distributions were restricted to be uniform distributions over the respective aggregates, and it was *not* assumed that $\Pi = APA$. In this context, $\tau(\chi)$, written as $\tau(\rho)$ for $\rho \in \Omega$ does indeed have the intuitive meaning of bounding the error caused by the aggregate ρ which corresponds to a group of states in the original chain.

The advantage of the more general bounds in Theorem 4 is that we can calculate error bounds for a wider set of possible aggregation schemes when compared to [5], as we are able to choose arbitrary values for π_0 , Π and A . In the following, we often distinguish two types of error:

$$\|e_k\|_1 \leq \underbrace{\|\pi_0^\top A - p_0^\top\|_1}_{\text{initial error}} + \underbrace{\sum_{j=0}^{k-1} \langle |\pi_j|, |\Pi A - AP| \cdot \mathbf{1}_n \rangle}_{\text{dynamic error}}$$

The initial error is independent of the transition matrix P and describes the error made by approximating p_0 with an aggregated initial distribution π_0 , while the dynamic error describes the error made by approximating the dynamic evolution of the Markov chain, given by the transition matrix P , with an aggregated dynamics, given by the step matrix Π .

3.2. Error bounds for CTMCs

We again want to give a bound on $\|e_t\|_1$, where we set $e_t = \tilde{p}_t - p_t$, as before. We have

$$\|e_t\|_1 = \|\pi_0^\top e^{\Theta t} A - p_0^\top e^{Q t}\|_1$$

In [5], error bounds for aggregated CTMCs were obtained by uniformizing the Markov chain and then applying the error bounds for DTMCs. We are now going to drop the detour via uniformization and show how to directly bound the approximation error of the transient distribution in continuous time with a very similar expression as in the discrete-time case via the error matrix $\Theta A - A Q$ (instead of $\Pi A - AP$ for DTMCs). The norm $\|\Theta A - A Q\|_\infty$ can now be interpreted as a rate of maximal error growth (instead of error growth per step). We again consider A , Θ and π_0 as arbitrary but fixed.

Theorem 5. *Let e_t be the error vector at time t , i.e.*

$$e_t^\top = \tilde{p}_t^\top - p_t^\top = \pi_0^\top e^{\Theta t} A - p_0^\top e^{Q t}$$

Then, we have that

$$\begin{aligned} &\|e_t\|_1 \text{ is absolutely continuous and differentiable for almost all } t, \text{ with} \\ &\frac{d}{dt} \|e_t\|_1 \leq \langle |\pi_t|, |\Theta A - A Q| \cdot \mathbf{1}_n \rangle \text{ for almost all } t, \text{ and} \\ &\limsup_{u \rightarrow 0} \frac{\|e_{t+u}\|_1 - \|e_t\|_1}{u} \leq \langle |\pi_t|, |\Theta A - A Q| \cdot \mathbf{1}_n \rangle \text{ for all } t \end{aligned}$$

and the following hold:

- (i) $\|e_t\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + \int_0^t \langle |\pi_u|, |\Theta A - A Q| \cdot \mathbf{1}_n \rangle du$
- (ii) $\|e_t\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + \|\pi_0\|_1 \cdot \|\Theta A - A Q\|_\infty \cdot \frac{e^{t\|\Theta\|_\infty} - 1}{\|\Theta\|_\infty}$ whenever $\|\Theta\|_\infty \neq 0$
- (iii) if Θ is a generator and π_0 is a probability distribution, then $\|e_t\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + t \cdot \|\Theta A - A Q\|_\infty$

Remark. The set of points in which $\|e_t\|_1$ is not differentiable corresponds to all points where one of the components of the vector e_t is zero (and where the derivative of the same component with respect to t is non-zero). This set has measure zero with respect to the Lebesgue measure.

Before being able to prove the above theorem, we need another lemma.

Lemma 6. Assume that $f : \mathbb{R} \rightarrow \mathbb{R}$ is differentiable in 0, and that $f(0) = 0$. Then

$$\limsup_{u \rightarrow 0} \frac{|f(u)|}{u} = \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{|f(u)|}{u} = |f'(0)|$$

Proof. If $f(0) = 0$, then we have by continuity of $|\cdot|$:

$$|f'(0)| = \left| \lim_{u \rightarrow 0} \frac{f(u)}{u} \right| = \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{|f(u)|}{|u|} = \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{|f(u)|}{u} = \limsup_{u \rightarrow 0} \frac{|f(u)|}{u} \quad \square$$

Proof of Theorem 5. First, note the following: every component of e_t is continuously differentiable in t , as both p_t and \tilde{p}_t are continuously differentiable with respect to t . Indeed, calculating the derivative of all components of e_t simultaneously, we get

$$\begin{aligned} \frac{d}{dt} (\tilde{p}_t^\top - p_t^\top) &= \frac{d}{dt} (\pi_0^\top e^{\Theta t} A - p_0^\top e^{Q t}) = \pi_0^\top e^{\Theta t} \Theta A - p_0^\top e^{Q t} Q = \pi_t^\top \Theta A - p_t^\top Q \\ \left\| \frac{d}{dt} (\tilde{p}_t^\top - p_t^\top) \right\|_1 &= \left\| \pi_t^\top \Theta A - p_t^\top Q \right\|_1 \leq \left\| \pi_t^\top \Theta A \right\|_1 + \left\| p_t^\top Q \right\|_1 \stackrel{\text{Lemma 1}}{\leq} \|\pi_0\|_1 \|e^{\Theta t}\|_\infty \|\Theta A\|_\infty + \|p_t\|_1 \|Q\|_\infty \\ &\stackrel{\text{Lemma 2}}{\leq} \|\pi_0\|_1 e^{t\|\Theta\|_\infty} \|\Theta A\|_\infty + \|Q\|_\infty \end{aligned} \tag{2}$$

As every component of e_t is continuously differentiable with bounded derivative, every component is absolutely continuous. Hence, $\|e_t\|_1$ is absolutely continuous as a sum of absolute values of absolutely continuous functions (this follows immediately from the definition of absolute continuity – see [12, beginning of Section 5.4 on page 108]) and therefore differentiable almost everywhere (see [12, Corollary 12 of Section 5.4 on page 109]) on any bounded time interval.

Noting that $e^{Q t} e^{Q u} = e^{Q(t+u)}$ by commutativity of $Q t$ and $Q u$, we see that, for $t, u \geq 0$,

$$\begin{aligned} \|e_{t+u}\|_1 &= \left\| \pi_0^\top e^{\Theta(t+u)} A - p_0^\top e^{Q(t+u)} \right\|_1 \\ &= \left\| \underbrace{(\pi_0^\top e^{\Theta t} A - p_0^\top e^{Q t})}_{e_t^\top} \underbrace{e^{Q u}}_{\text{stochastic}} + \pi_0^\top e^{\Theta t} (e^{\Theta u} A - A e^{Q u}) \right\|_1 \\ &\stackrel{\Delta\text{-inequ. \& Lemma 1}}{\leq} \|e_t\|_1 + \left\| \pi_t^\top (e^{\Theta u} A - A e^{Q u}) \right\|_1 \stackrel{\text{Lemma 1}}{\leq} \left\langle \|e_t\|_1 + \langle |\pi_t|, |e^{\Theta u} A - A e^{Q u}| \cdot \mathbf{1}_n \rangle \right. \\ &\quad \left. \left\| e_t\|_1 + \|\pi_t\|_1 \cdot \|e^{\Theta u} A - A e^{Q u}\|_\infty \right. \end{aligned} \tag{3}$$

To obtain a more precise bound, we will continue by using the first inequality derived above. Note: we have that (the matrices $e^{Q u}$ and Q commute)

$$\frac{d}{du} (e^{\Theta u} A - A e^{Q u}) = e^{\Theta u} \Theta A - A e^{Q u} Q \stackrel{u=0}{=} \Theta A - A Q$$

Hence:

$$\begin{aligned} \|e_{t+u}\|_1 - \|e_t\|_1 &\leq \langle |\pi_t|, |e^{\Theta u} A - A e^{Q u}| \cdot \mathbf{1}_n \rangle \\ \implies \limsup_{u \rightarrow 0} \frac{\|e_{t+u}\|_1 - \|e_t\|_1}{u} &= \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{\|e_{t+u}\|_1 - \|e_t\|_1}{u} \\ &\leq \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{\langle |\pi_t|, |e^{\Theta u} A - A e^{Q u}| \cdot \mathbf{1}_n \rangle}{u} = \lim_{u > 0} \sum_{\chi=1}^m |\pi_t(\chi)| \cdot \sum_{i=1}^n \frac{|(e^{\Theta u} A - A e^{Q u})(\chi, i)|}{u} \\ &= \sum_{\chi=1}^m |\pi_t(\chi)| \cdot \sum_{i=1}^n \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{|(e^{\Theta u} A - A e^{Q u})(\chi, i)|}{u} \stackrel{\text{Lemma 6}}{=} \sum_{\chi=1}^m |\pi_t(\chi)| \cdot \sum_{i=1}^n |(\Theta A - A Q)(\chi, i)| \\ &= \langle |\pi_t|, |\Theta A - A Q| \cdot \mathbf{1}_n \rangle \end{aligned}$$

This proves the bounds on the derivative given in Theorems 5 and (i) follows immediately. Going back to (3), we can also derive a less precise bound using the lower inequality in (3): we have that

$$\begin{aligned} \limsup_{u \rightarrow 0} \frac{\|e_{t+u}\|_1 - \|e_t\|_1}{u} &= \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{\|e_{t+u}\|_1 - \|e_t\|_1}{u} \\ &\leq \lim_{u > 0} \|\pi_t\|_1 \cdot \frac{\|e^{\Theta u} A - A e^{Q u}\|_\infty}{u} = \lim_{u > 0} \|\pi_t\|_1 \cdot \max_{\chi=1, \dots, m} \sum_{i=1}^n \frac{|(e^{\Theta u} A - A e^{Q u})(\chi, i)|}{u} \end{aligned}$$

$$\begin{aligned}
 &= \|\pi_t\|_1 \cdot \max_{\chi=1,\dots,m} \sum_{i=1}^n \lim_{\substack{u \rightarrow 0 \\ u > 0}} \frac{|(e^{\Theta u} A - A e^{Q u})(\chi, i)|}{u} \\
 &\stackrel{\text{Lemma 6}}{=} \|\pi_t\|_1 \cdot \max_{\chi=1,\dots,m} \sum_{i=1}^n |(\Theta A - A Q)(\chi, i)| = \|\pi_t\|_1 \cdot \|\Theta A - A Q\|_\infty
 \end{aligned}$$

Therefore,

$$\begin{aligned}
 \|e_t\|_1 &\leq \|\pi_0^\top A - p_0^\top\|_1 + \int_0^t \|\pi_u\|_1 \cdot \|\Theta A - A Q\|_\infty \, du \\
 &\leq \|\pi_0^\top A - p_0^\top\|_1 + \|\pi_0\|_1 \cdot \left(\int_0^t \|e^{\Theta u}\|_\infty \, du \right) \cdot \|\Theta A - A Q\|_\infty
 \end{aligned}$$

where we used that $\|\pi_t\|_1 \stackrel{\text{Lemma 1}}{\leq} \|\pi_0\|_1 \cdot \|e^{\Theta t}\|_\infty$. If Θ is a generator and if π_0 is a probability distribution, this inequality collapses to

$$\|e_t\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1 + t \cdot \|\Theta A - A Q\|_\infty$$

which proves (iii). To prove (ii), we note that

$$\int_0^t \|e^{\Theta u}\|_\infty \, du \stackrel{\text{Lemma 2}}{\leq} \int_0^t e^{u\|\Theta\|_\infty} \, du = \left[\frac{e^{u\|\Theta\|_\infty}}{\|\Theta\|_\infty} \right]_{u=0}^t = \frac{e^{t\|\Theta\|_\infty} - 1}{\|\Theta\|_\infty} \quad \square$$

3.3. When is the error bound zero?

Corollary 7. Given a DTMC, an arbitrary disaggregation matrix A , an arbitrary aggregated step matrix Π , and an arbitrary initial vector π_0 , we have the following. If $\Pi A = AP$, then $\|\tilde{p}_k - p_k\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1$.

Analogously, for a given CTMC, disaggregation matrix A , aggregated evolution matrix Θ , and initial vector p_0 , we have: if $\Theta A = A Q$, then $\|\tilde{p}_t - p_t\|_1 \leq \|\pi_0^\top A - p_0^\top\|_1$.

Proof. This follows directly from Theorems 4 and 5. \square

Corollary 7 motivates the following definition:

Definition 8. We call an aggregation of a DTMC, given by a disaggregation matrix A , an aggregated step matrix Π and an aggregated initial vector π_0 **dynamic-exact** if $\Pi A = AP$. Analogously, we call an aggregation of a CTMC dynamic-exact if $\Theta A = A Q$ where Θ is the aggregated evolution matrix.

If we further have that $\pi_0^\top A = p_0^\top$, then we call the aggregation **exact**.

If A , Π and π_0 are an exact aggregation, then $\tilde{p}_k = p_k$ for all k . This was already shown in Corollary 7, but can also be proven easily by induction. Indeed, $\Pi A = AP$ implies $AP^k = AP P^{k-1} = \Pi A P P^{k-2} = \dots = \Pi^k A$. Hence, if $\pi_0^\top A = p_0^\top$, then we have

$$p_k^\top = p_0^\top P^k = \pi_0^\top A P^k = \pi_0^\top \Pi^k A = \pi_k^\top A = \tilde{p}_k^\top \tag{4}$$

If the aggregation is only dynamic-exact, then it only holds that $\tilde{p}_k^\top = \tilde{p}_0^\top P^k$ as

$$\tilde{p}_0^\top P^k = \pi_0^\top A P^k = \pi_0^\top \Pi^k A = \pi_k^\top A = \tilde{p}_k^\top$$

Note that in this case, we might have $\tilde{p}_0 \neq p_0$. This is the reason for the term ‘‘dynamic-exact’’: the step dynamics are correctly represented by the aggregation, but the initial distribution might be wrongly approximated by the initial vector \tilde{p}_0 . The corresponding statements also hold for CTMCs.

Proposition 9. Consider an aggregation of a DTMC which is dynamic-exact, i.e. $\Pi A = AP$. Further assume that either $\pi_0^\top A \geq p_0^\top$ or $\pi_0^\top A \leq p_0^\top$, where the inequalities should hold entry-wise. Then:

$$\left\| \pi_0^\top \Pi^k A - p_0^\top P^k \right\|_1 \stackrel{\text{def.}}{=} \|e_k\|_1 = \left\| \pi_0^\top A - p_0^\top \right\|_1$$

Furthermore, if $\pi_0^\top A \geq p_0^\top$, then $\tilde{p}_k^\top = \pi_0^\top \Pi^k A \geq p_k^\top$ for all k (and the same holds for \leq).

Analogously, for a dynamic-exact aggregation of a CTMC with either $\pi_0^\top A \geq p_0^\top$ or $\pi_0^\top A \leq p_0^\top$, we have

$$\left\| \pi_0^\top e^{\Theta t} A - p_0^\top e^{Q t} \right\|_1 \stackrel{\text{def.}}{=} \|e_t\|_1 = \left\| \pi_0^\top A - p_0^\top \right\|_1$$

and it also holds that $\tilde{p}_t^\top = \pi_0^\top e^{\Theta t} A \geq p_t^\top$ for all t if $\pi_0^\top A \geq p_0^\top$ (and the same for \leq).

Proof. Note that

$$\begin{aligned}
 \pi_0^\top \Pi^k A - p_0^\top P^k &= \underbrace{\pi_0^\top \Pi^k A - \pi_0^\top A P^k}_{=0 \text{ as } \Pi A = AP} + \pi_0^\top A P^k - p_0^\top P^k = (\pi_0^\top A - p_0^\top) P^k
 \end{aligned}$$

which proves the claim for DTMCs as P^k is stochastic and therefore preserves the $\|\cdot\|_1$ -norms of non-negative and non-positive vectors applied from the left as well as the non-negativity or non-positivity itself. For CTMCs, we have

$$\begin{aligned} \pi_0^\top e^{\Theta t} A - p_0^\top e^{Q t} &= \pi_0^\top e^{\Theta t} A - \pi_0^\top A e^{Q t} + \pi_0^\top A e^{Q t} - p_0^\top e^{Q t} \\ &= \underbrace{\pi_0^\top \sum_{k=0}^{\infty} \frac{\Theta^k A - A Q^k}{k!}}_{=0 \text{ as } \Theta A = A Q} + (\pi_0^\top A - p_0^\top) e^{Q t} \end{aligned}$$

which proves the claim for CTMCs as $e^{Q t}$ is stochastic as well. \square

The above Proposition 9 allows us to find lower and upper bounds for the transient distributions by choosing appropriate values for π_0 which result in lower or upper bounds of p_0 (in the sense that $\pi_0^\top A \geq p_0^\top$ or $\pi_0^\top A \leq p_0^\top$), if we have a dynamic-exact aggregation.

The condition $\Pi A = A P$ has appeared in the literature before in connection with Markov chains, but always in more restricted contexts, in particular in the context of weighted state space partitioning as defined in Section 2.2 (where the state space S of the original chain is partitioned into aggregates by Ω and each state s of the original chain is assigned weight $\alpha(s)$ within its aggregate). Indeed, Eq. (4) on page 135 of [2] states that, if $\Pi = A P A$, then $\Pi A = A P$ implies weak lumpability of the DTMC. A DTMC is called weakly lumpable for a given partition Ω if there exists an initial distribution p_0 such that the process Y_k , defined by $Y_k = \sigma \in \Omega \iff X_k \in \sigma$, is a Markov chain. For such an initial distribution, the probabilities $\pi_k(\sigma)$ are then exactly equal to $\mathbb{P}[Y_k = \sigma] = \mathbb{P}[X_k \in \sigma]$. However, the concept of weak lumpability makes no statement about whether the probability $\mathbb{P}[X_k = s]$ for $s \in \sigma$ can be accurately derived from the knowledge of $\mathbb{P}[X_k \in \sigma]$.

Again under the condition that $\Pi = A P A$, [13, Definition 2.2] defined the matrix P to be A -lumpable if $\Pi A = A P$. In the subsequent remarks, [13] then noted that, given π_k , an exact recovery of the probabilities $p_k(s)$ is possible provided that the initial distribution p_0 is compatible with the α distributions. Actually, [13, equation (2.4)] corresponds almost exactly to (4).

Furthermore, an exact aggregation is called backward bisimulation of type 2 in [14, Definition 4.3], yet again if $\Pi = A P A$. [14] already remarks that the coarsest exact aggregation cannot be found by the usual partition refinement approach (we will revisit the partition refinement technique later in Section 5.3). Therefore, efficiently finding partitions Ω (and distributions α) which are an exact aggregation for a given DTMC or CTMC is currently still an open research problem, to the best of our knowledge.

3.4. Error bound for the approximated stationary distribution

This paper focuses on error bounds for the approximated transient distributions of a Markov chain. However, we will briefly cover a similar error bound for how well we can approximate the stationary distribution with an aggregated chain. For simplicity, we will assume in this section that the DTMC given by P and the CTMC given by Q give rise to a unique stationary distribution p . We first consider the discrete-time case.

As we allow arbitrary matrices for the aggregated step matrix Π , we define an approximation for the stationary distribution p of the original chain as follows. A stationary vector π of Π simply is a left eigenvector of the matrix Π with eigenvalue 1, i.e. it holds that $\pi^\top \Pi = \pi^\top$. Given such a stationary vector, we can approximate p^\top by the vector $\pi^\top A$, which is very similar to the approximation $\tilde{p}_k^\top = \pi_k^\top A$ of p_k . We then get

$$\left\| \pi^\top A P - \pi^\top A \right\|_1 = \left\| \pi^\top A P - \pi^\top \Pi A \right\|_1 \stackrel{\text{Lemma 1}}{\leq} \|\pi\|_1 \|\Pi A - A P\|_\infty$$

The leftmost term gives a measure of how far the approximated stationary distribution $\pi^\top A$ is from being stationary for the matrix P . Note that depending on our choice of Π , there might not exist a left eigenvector with eigenvalue 1. In the special case where Π and A are stochastic matrices, the existence of π is guaranteed, π is an actual stationary distribution for Π , and we will also have that $\pi^\top A$ is a probability distribution. $\|\Pi A - A P\|_\infty$ gives us some sort of measure of how far this probability distribution is from being stationary. Hence, we can see $\|\Pi A - A P\|_\infty$ as not only capturing how well the aggregated chain reflects the original transient behavior, but also capturing how well we can approximate the stationary behavior.

If Π does not have a left eigenvector with eigenvalue 1, we can consider the case where we only have the following approximate statement: $\|\pi^\top \Pi - \pi^\top\|_1 \approx 0$. In this case,

$$\left\| \pi^\top A P - \pi^\top A \right\|_1 = \left\| \pi^\top A P - \pi^\top \Pi A + (\pi^\top \Pi - \pi^\top) A \right\|_1 \leq \|\pi\|_1 \|\Pi A - A P\|_\infty + \left\| \pi^\top \Pi - \pi^\top \right\|_1 \|A\|_\infty$$

For stochastic A and a probability distribution π , this inequality reduces to

$$\left\| \pi^\top A P - \pi^\top A \right\|_1 \leq \|\Pi A - A P\|_\infty + \left\| \pi^\top \Pi - \pi^\top \right\|_1$$

For CTMCs, we have the corresponding result: if π is a left eigenvector of Θ with eigenvalue 0, i.e. $\pi^\top \Theta = 0$, then

$$\left\| \pi^\top A Q \right\|_1 = \left\| \pi^\top A Q - \pi^\top \Theta A \right\|_1 \stackrel{\text{Lemma 1}}{\leq} \|\pi\|_1 \|\Theta A - A Q\|_\infty$$

and if we only have that $\|\pi^\top \Theta\|_1 \approx 0$, then

$$\left\| \pi^\top A Q \right\|_1 \leq \|\pi\|_1 \|\Theta A - A Q\|_\infty + \left\| \pi^\top \Theta \right\|_1 \|A\|_\infty$$

Corollary 10. Consider a DTMC with transition matrix P , and aggregated step matrix Π , or a CTMC with generator matrix Q , and aggregated evolution matrix Θ . Let A be the disaggregation matrix and let $\pi \in \mathbb{R}^m$ be an approximate stationary vector for the aggregated model (i.e. $\pi^\top \Pi \approx \pi^\top$ or $\pi^\top \Theta \approx 0$). Then, the following holds for $\pi^\top A$, which is an approximation of the stationary distribution of the original chain:

$$\begin{aligned} \|\pi^\top AP - \pi^\top A\|_1 &\leq \|\pi\|_1 \|\Pi A - AP\|_\infty + \|\pi^\top \Pi - \pi^\top\|_1 \|A\|_\infty \quad \text{and} \\ \|\pi^\top AQ\|_1 &\leq \|\pi\|_1 \|\Theta A - AQ\|_\infty + \|\pi^\top \Theta\|_1 \|A\|_\infty \end{aligned}$$

If A is stochastic and π is a probability distribution, then

$$\|\pi^\top AP - \pi^\top A\|_1 \leq \|\Pi A - AP\|_\infty + \|\pi^\top \Pi - \pi^\top\|_1 \quad \text{and} \quad \|\pi^\top AQ\|_1 \leq \|\Theta A - AQ\|_\infty + \|\pi^\top \Theta\|_1$$

Remark. The bounds given above only measure how close to stationarity $\pi^\top A$ is, the distance of $\pi^\top A$ to the actual stationary distribution p of the original chain can be both smaller or larger. Indeed, as shown by the following examples, it is both possible that $\|p^\top - \pi^\top A\|_1 < \|\Pi A - AP\|_\infty$ and that $\|p^\top - \pi^\top A\|_1 > \|\Pi A - AP\|_\infty$ if p is a stationary distribution of P and if π is a stationary distribution for the stochastic matrix Π (so we actually have $\pi^\top \Pi = \pi^\top$ and not just $\pi^\top \Pi \approx \pi^\top$, and the reduced model is a Markov chain as Π is stochastic). We look at the following DTMC with stationary measure p :

$$P = \frac{1}{4} \cdot \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix} \implies p = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^\top$$

- To see that $\|p^\top - \pi^\top A\|_1 < \|\Pi A - AP\|_\infty$ is possible, consider the following (here we used weighted state space partitioning and $\Pi = APA$):

$$\begin{aligned} A &= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \Pi = \frac{1}{8} \cdot \begin{pmatrix} 5 & 3 \\ 6 & 2 \end{pmatrix} \implies \pi = \left(\frac{2}{3}, \frac{1}{3}\right)^\top, \quad \pi^\top A = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) \\ \implies \|p^\top - \pi^\top A\|_1 &= 0 < \frac{1}{4} = \|\Pi A - AP\|_\infty \end{aligned}$$

- To see that $\|p^\top - \pi^\top A\|_1 > \|\Pi A - AP\|_\infty$ is possible as well, consider the following (note that the only difference to the previous example is the choice of A):

$$\begin{aligned} A &= \begin{pmatrix} \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \Pi = \frac{1}{16} \cdot \begin{pmatrix} 9 & 7 \\ 12 & 4 \end{pmatrix} \implies \pi = \left(\frac{12}{19}, \frac{7}{19}\right)^\top, \quad \pi^\top A = \left(\frac{9}{19}, \frac{3}{19}, \frac{7}{19}\right) \\ \implies \|p^\top - \pi^\top A\|_1 &= \frac{20}{57} > 0.35 > 0.344 > \frac{11}{32} = \|\Pi A - AP\|_\infty \end{aligned}$$

Thus, $\|\Pi A - AP\|_\infty$ is not a bound on how far off the approximated stationary distribution, obtained via the aggregated chain, is from the actual stationary distribution of the chain. Only the result from [Corollary 10](#) holds: $\|\Pi A - AP\|_\infty$ is a measure of how far from stationarity $\pi^\top A$ is, in the sense that $\|\pi^\top AP - \pi^\top A\|_1 \leq \|\Pi A - AP\|_\infty$.

3.5. Tightness of the error bound

We now want to show that the bounds given in [Theorems 4](#) and [5](#) are tight in the following sense: if P , Π and A are fixed and $\|\Pi A - AP\|_\infty > 0$, then we can find initial vectors p_0 and π_0 such that the error bounds are actually achieved. We show this only for matrices A having non-negative entries.

Theorem 11. Consider an aggregation of a DTMC, given by a disaggregation matrix A , and an aggregated step matrix Π . Note that we do not fix π_0 yet. If every entry of A is non-negative, if every row of A contains at least one strictly positive entry, and if $\|\Pi A - AP\|_\infty > 0$, then the following holds: there exists an initial distribution p_0 and an aggregated initial vector π_0 such that $\pi_0^\top A = p_0^\top$ and such that we have $\|\tilde{p}_1 - p_1\|_1 = \|\pi_0\|_1 \cdot \|\Pi A - AP\|_\infty$.

For an aggregation of a CTMC with disaggregation matrix A and aggregated evolution matrix Θ , and under the same condition on A as above, we have the analog statement: if $\|\Theta A - AQ\|_\infty > 0$, then there exist p_0 and π_0 with $\pi_0^\top A = p_0^\top$ and such that $\lim_{t \rightarrow 0, t > 0} \frac{1}{t} \|\tilde{p}_t - p_t\|_1 = \|\pi_0\|_1 \cdot \|\Theta A - AQ\|_\infty$.

Proof. Let

$$\chi^* := \arg \max_{\chi=1, \dots, m} \sum_{i=1}^n |(\Pi A - AP)(\chi, i)| \quad \text{or, for CTMCs,} \quad \chi^* := \arg \max_{\chi=1, \dots, m} \sum_{i=1}^n |(\Theta A - AQ)(\chi, i)|$$

If there are multiple possible choices for χ^* , we may choose any of them. We now set

$$\pi_0(\chi) := \begin{cases} 0 & \text{if } \chi \neq \chi^* \\ \frac{1}{\sum_{i=1}^n |A(\chi^*, i)|} & \text{if } \chi = \chi^*, \end{cases} \quad p_0^\top := \pi_0^\top A$$

Note that setting p_0 as above implies that p_0 is a probability distribution: every entry is non-negative, as both π_0 and A contain only non-negative entries (the latter by assumption), and since

$$\sum_{i=1}^n p_0(i) = \sum_{i=1}^n \sum_{\chi=1}^m \pi_0(\chi) A(\chi, i) = \sum_{i=1}^n \frac{1}{\sum_{j=1}^n |A(\chi^*, j)|} \cdot A(\chi^*, i) = 1$$

As we assumed that every row of A contains at least one strictly positive entry, we also know that $\sum_{i=1}^n |A(\chi^*, i)| > 0$, ensuring that we do not divide by zero. But now, in the discrete-time case,

$$\begin{aligned} \|\tilde{p}_1 - p_1\|_1 &= \|\pi_0^\top \Pi A - p_0^\top P\|_1 = \|\pi_0^\top \Pi A - \pi_0^\top A P\|_1 = \|\pi_0^\top (\Pi A - A P)\|_1 = \pi_0(\chi^*) \sum_{i=1}^n |(\Pi A - A P)(\chi^*, i)| \\ &= \|\pi_0\|_1 \cdot \|\Pi A - A P\|_\infty \end{aligned}$$

where the last equality holds by choice of χ^* . Looking at the continuous-time case, we find that (with the choice of π_0 and p_0 as given above),

$$\frac{d}{dt} \Big|_{t=0} (\tilde{p}_t^\top - p_t^\top) \stackrel{(2)}{=} \pi_0^\top \Theta A - p_0^\top Q = \pi_0^\top \Theta A - \pi_0^\top A Q = \pi_0^\top (\Theta A - A Q)$$

Hence, noting that $\|\tilde{p}_0 - p_0\|_1 = 0$, we obtain

$$\begin{aligned} \lim_{t \rightarrow 0} \frac{1}{t} \|\tilde{p}_t - p_t\|_1 &= \sum_{i=1}^n \lim_{t \rightarrow 0} \frac{|(\tilde{p}_t - p_t)(i)|}{t} \stackrel{\text{Lemma 6}}{=} \sum_{i=1}^n |(\pi_0^\top (\Theta A - A Q))(i)| = \|\pi_0^\top (\Theta A - A Q)\|_1 \\ &= \dots \text{ (as in the discrete-time case)} = \|\pi_0\|_1 \cdot \|\Theta A - A Q\|_\infty \quad \square \end{aligned}$$

3.6. Reduction to arbitrarily low dimension

As shown in the following proposition, we can actually find dynamic-exact aggregations of any arbitrarily low dimension.

Proposition 12. Consider a DTMC on a state space of size n . For every $m < n$, there either exist matrices $A \in \mathbb{R}^{m \times n}$ and $\Pi \in \mathbb{R}^{m \times m}$ or matrices $A \in \mathbb{R}^{(m+1) \times n}$ and $\Pi \in \mathbb{R}^{(m+1) \times (m+1)}$ where A has full rank and such that $\Pi A = A P$.

Analogously, for every CTMC with a state space of size n and for every $m < n$, we can find $A \in \mathbb{R}^{m \times n}$ and $\Theta \in \mathbb{R}^{m \times m}$ or $A \in \mathbb{R}^{(m+1) \times n}$ and $\Theta \in \mathbb{R}^{(m+1) \times (m+1)}$ where A has full rank and such that $\Theta A = A Q$.

Proof. This follows from the existence of the real Schur decomposition of P^\top (see [11, Theorem 7.4.1]). Indeed, there exists an orthogonal matrix $U \in \mathbb{R}^{n \times n}$ and a quasi upper triangular matrix $T \in \mathbb{R}^{n \times n}$ such that $P^\top = U T U^\top$. Here, quasi upper triangular means that

$$T = \begin{pmatrix} T_{11} & T_{12} & \dots & T_{1k} \\ 0 & T_{22} & \dots & T_{2k} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & T_{kk} \end{pmatrix} \text{ where } T_{ii} \in \mathbb{R}^{1 \times 1} \text{ or } T_{ii} \in \mathbb{R}^{2 \times 2} \quad \forall i \in \{1, \dots, k\}$$

so T is almost upper triangular with the exception that there might be non-zero values on the diagonal one below the main diagonal. The 1×1 blocks on the diagonal correspond to the real eigenvalues of the matrix P^\top , and the 2×2 blocks to the complex eigenvalues (see [11, Section 7.4.1] for details). Hence, we have that

$$P = U T^\top U^\top \implies U^\top P = \begin{pmatrix} T_{11}^\top & 0 & \dots & 0 \\ T_{12}^\top & T_{22}^\top & \dots & 0 \\ \vdots & & \ddots & \vdots \\ T_{1k}^\top & T_{2k}^\top & \dots & T_{kk}^\top \end{pmatrix} U^\top$$

We can now simply choose Π to correspond to the first m rows and first m columns of T^\top if we do not cut one of the T_{ij} blocks in half by doing this. Otherwise, we can choose Π to correspond to the first $m + 1$ rows and first $m + 1$ columns of T^\top . If we then choose A to be the first m or $m + 1$ rows of U^\top (and all columns), it does hold that $A P = \Pi A$, as required. Note that A will have full rank as the rows of A are orthonormal vectors. The same method can be applied to obtain A and Θ for a generator matrix Q . \square

Remark. In Proposition 12, it was stated that A can be chosen to have full rank. Π can actually also be chosen to have full rank if it is calculated using the Schur decomposition, and if the dimension of the reduced state space is at most the rank of P (this can be seen easily as the submatrix consisting of the first m rows and columns of T^\top in the proof of Proposition 12 will have full rank if

Table 1

Overview of error bounds.

DTMCs	CTMCs
π_0 probability distribution, Π and A stochastic, π stationary probability distribution for Π	π_0 probability distribution, Θ generator, A stochastic, π stationary probability distribution for Θ
Transient error bounds (precise) $\ e_k\ _1 \leq \ \pi_0^T A - p_0^T\ _1 + \sum_{j=0}^{k-1} \langle \pi_j , \Pi A - AP \cdot \mathbf{1}_n \rangle$	$\ e_t\ _1 \leq \ \pi_0^T A - p_0^T\ _1 + \int_0^t \langle \pi_u , \Theta A - AQ \cdot \mathbf{1}_n \rangle du$
Transient error bounds (imprecise) $\ e_k\ _1 \leq \ \pi_0^T A - p_0^T\ _1 + k \cdot \ \Pi A - AP\ _\infty$	$\ e_t\ _1 \leq \ \pi_0^T A - p_0^T\ _1 + t \cdot \ \Theta A - AQ\ _\infty$
Stationary error bounds $\ \pi^T AP - \pi^T A\ _1 \leq \ \Pi A - AP\ _\infty$	$\ \pi^T A Q\ _1 \leq \ \Theta A - A Q\ _\infty$

the blocks on the diagonal correspond to non-zero eigenvalues, where we note that the order in which the eigenvalue blocks appear on the diagonal can be chosen arbitrarily in the Schur decomposition with an appropriate algorithm).

It makes sense to choose full-rank matrices — otherwise, there would still be redundant information in the reduced model. Indeed, if we consider arbitrary matrices A and Π which do not necessarily have full rank, the following trivial aggregation will always be dynamic-exact (for simplicity, we assume that P gives rise to a unique stationary distribution p):

$$A = \begin{pmatrix} - & p^T & - \\ & \vdots & \\ - & p^T & - \end{pmatrix}, \quad \Pi = \begin{pmatrix} \frac{1}{m} & \dots & \frac{1}{m} \\ \vdots & \ddots & \vdots \\ \frac{1}{m} & \dots & \frac{1}{m} \end{pmatrix} \quad \text{where } p \text{ is the stationary distribution of } P$$

In this case, $AP = A$ by choice of A , and it is easy to see that we also have $\Pi A = A$. In fact, choosing $A = p^T$ and $\Pi = (1) \in \mathbb{R}^{1 \times 1}$ always results in a dynamic-exact aggregation with only a single aggregate state. This aggregation does furthermore respect the restrictions of weighted state space partitioning, so we can always find a dynamic-exact aggregation with $m = 1$ aggregate when using weighted state space partitioning.

At this point, one might ask why we should consider dynamic-exact aggregations with more than a single aggregate or even why we should consider aggregations which are not dynamic-exact. The biggest disadvantage of a dynamic-exact aggregation with a low number of aggregates is that the initial error $\|\pi_0^T A - p_0^T\|_1$ increases as m decreases, generally speaking, since with decreasing m , we have fewer degrees of freedom for choosing our approximate initial distribution. The other disadvantage is the computational cost of finding the Schur decomposition or the stationary distribution. It takes $\mathcal{O}(n^3)$ time to perform a Schur decomposition (see [11, Section 7.4]), which can quickly get out of hand for large state spaces where we are actually most interested in reducing the state space. As for finding the stationary distribution: if we are actually interested in the transient distributions, then the dynamic-exact aggregation with $A = p^T$ and $\Pi = (1) \in \mathbb{R}^{1 \times 1}$ will not be helpful at all, and for stationary analysis, we are interested in aggregation for speeding up the computation of the stationary distribution, so this dynamic-exact aggregation is obviously not helpful, either.

While we can, in theory, find dynamic-exact aggregations with reduced state spaces of arbitrarily low dimension, this is not possible for exact aggregations. Consider a reduction to a state space of dimension $m = 1$. The matrix Π will reduce to a scalar value, and the matrix A to a single row. For $\Pi A = AP$ to hold, we would therefore need that the row of A is a left eigenvector of P . If we also require that $\pi_0^T A = p_0^T$ (note that π_0 also reduces to a scalar value for $m = 1$), which we need for an exact aggregation, then we find that the row of A must furthermore be a multiple of the initial distribution p_0 . But a left eigenvector of P which is a multiple of a probability distribution necessarily is a stationary distribution of the Markov chain, if normed appropriately. Hence, an exact aggregation to a state space of dimension 1 exists if, and only if, p_0 is a stationary distribution of the Markov chain.

3.7. Summary of results

Table 1 lists the most important error bounds (from Theorems 4, 5 and Corollary 10) for the case where Π and A are stochastic and Θ is a generator, which is most relevant in practical applications. $\|\Pi A - AP\|_\infty$ and $\|\Theta A - A Q\|_\infty$ can be seen as measuring how well the aggregated model reflects the transient and stationary behavior of the full Markov chain, and $\|\pi_0^T A - p_0^T\|_1$ measures the error made when approximating the initial distribution, which is relevant for the faithfulness of the approximation of the transient behavior as well.

4. Lumpability and aggregatability

In this section, we will put the error bounds presented in the previous section in context by comparing them with concepts from the related literature. As most publications focus on weighted state space partitioning as defined in Section 2.2, we will focus on this special case in this section, i.e. the state space S of the original chain is partitioned into aggregates by Ω , each state s of the original chain is assigned weight $\alpha(s)$ within its aggregate, we set $\Pi = APA$, $\Theta = AQA$, $\pi_0^T = p_0^T A$, and the rows of A are the α distributions. From now on, we will index P by pairs of states $r, s \in S$, i.e. we will write $P(r, s)$, and we will index Π by pairs of aggregates $\rho, \sigma \in \Omega$, i.e. $\Pi(\rho, \sigma)$.

4.1. An alternative characterization of dynamic-exactness

We first compare the notion of exactness and dynamic-exactness as defined in [Definition 8](#) with a similar notion from [\[10\]](#) for aggregations via weighted state space partitioning. Recall the following notation: $\omega : S \rightarrow \Omega$ maps a state s to its aggregate $\omega(s)$. [\[10, Theorem 2\]](#) and [\[10, Theorem 9\]](#) state that the distributions \tilde{p}_k (respectively \tilde{p}_l) are equal to p_k (respectively p_l) under the following conditions:

- (i) $\forall s, s' \in S$ s.t. $\omega(s) = \omega(s') : \forall \rho \in \Omega : \frac{\sum_{r \in \rho} \alpha(r)P(r, s)}{\alpha(s)} = \frac{\sum_{r \in \rho} \alpha(r)P(r, s')}{\alpha(s')}$
- (ii) the initial distribution p_0 is compatible with the α distributions (compare with [Definition 3](#); this is called p_0 respects the α distributions in [\[10\]](#)). Recall that this is the case when $p_0^T \Lambda A = p_0^T$, or $\pi_0^T A = p_0^T$ as $p_0^T \Lambda = \pi_0^T$ if we consider weighted state space partitioning.

The next proposition, [Proposition 13](#), shows that (i) is equivalent to $\| \Pi A - AP \|_\infty = 0$ (respectively $\| \Theta A - A Q \|_\infty = 0$) and thus by [Definition 8](#) equivalent to dynamic-exactness. It follows that conditions (i) and (ii) together are equivalent to exactness, and hence imply (see [Corollary 7](#)) that the transient distributions \tilde{p}_k (respectively \tilde{p}_l) agree with p_k (respectively p_l), which is exactly the statement of [\[10, Theorem 2\]](#) and [\[10, Theorem 9\]](#), but obtained via a different proof.

Proposition 13. *Given an irreducible DTMC, a partition Ω of its state space, and arbitrary probability distributions α_σ with support on $\sigma \in \Omega$, let $\Pi = APA$ (where the row vectors of A correspond to the distributions α_σ). Then:*

$$\begin{aligned} & \| \Pi A - AP \|_\infty = 0 \\ & \iff \\ & \forall s \in S : \alpha(s) > 0 \text{ and} \\ & \forall s, s' \in S \text{ s.t. } \omega(s) = \omega(s') : \forall \rho \in \Omega : \frac{\sum_{r \in \rho} \alpha(r)P(r, s)}{\alpha(s)} = \frac{\sum_{r \in \rho} \alpha(r)P(r, s')}{\alpha(s')} \end{aligned}$$

Furthermore, the assumption $\Pi = APA$ is only needed for the \iff direction, the other direction \implies in the equation above holds for an arbitrary choice of Π . The same statement holds for irreducible CTMCs with Π replaced by Θ and P replaced by Q .

Proof. We have that

$$\begin{aligned} \| \Pi A - AP \|_\infty = 0 & \iff \forall \rho \in \Omega : \forall s \in S : (\Pi A)(\rho, s) = (AP)(\rho, s) \\ & \iff \forall \rho \in \Omega : \forall s \in S : \sum_{\sigma \in \Omega} \Pi(\rho, \sigma) A(\sigma, s) = \sum_{r \in S} A(\rho, r) P(r, s) \\ & \iff \forall \rho \in \Omega : \forall s \in S : \Pi(\rho, \omega(s)) \alpha(s) = \sum_{r \in \rho} \alpha(r) P(r, s) \\ & \iff \forall \rho, \sigma \in \Omega : \forall s \in \sigma : \alpha(s) \Pi(\rho, \sigma) = \sum_{r \in \rho} \alpha(r) P(r, s) \end{aligned} \tag{5}$$

We first show by contradiction: $\| \Pi A - AP \|_\infty = 0$ implies $\forall s \in S : \alpha(s) > 0$. Call $S_0 := \{s_0 \in S : \alpha(s_0) = 0\} \neq S$ (we have $S_0 \neq S$ because the α_σ are probability distributions, so there must be at least one state s_{-0} with $\alpha(s_{-0}) > 0$). If $S_0 \neq \emptyset$, then, by our assumption of irreducibility, there must be some state $s \in S_0$ and some state $r \in S \setminus S_0$ (hence $\alpha(r) > 0$) with $P(r, s) > 0$. In particular, we have that $\sum_{r' \in \omega(r)} \alpha(r') P(r', s) > 0$ while $\alpha(s) \Pi(\omega(r), \omega(s)) = 0$. Thus, since the last statement of (5) does not hold in this case (it is violated for $\rho = \omega(r)$, $\sigma = \omega(s)$ and s), it follows that $\| \Pi A - AP \|_\infty > 0$. $S_0 \neq \emptyset$ is thus impossible when $\| \Pi A - AP \|_\infty = 0$, and hence, writing \circledast for $\forall s \in S : \alpha(s) > 0$:

$$\begin{aligned} \| \Pi A - AP \|_\infty = 0 & \iff \circledast \text{ and } \forall \rho, \sigma \in \Omega : \forall s \in \sigma : \alpha(s) \Pi(\rho, \sigma) = \sum_{r \in \rho} \alpha(r) P(r, s) \\ & \iff \circledast \text{ and } \forall \rho, \sigma \in \Omega : \forall s \in \sigma : \Pi(\rho, \sigma) = \frac{\sum_{r \in \rho} \alpha(r) P(r, s)}{\alpha(s)} \\ & \iff \circledast \text{ and } \forall \rho, \sigma \in \Omega : \forall s, s' \in \sigma : \frac{\sum_{r \in \rho} \alpha(r) P(r, s)}{\alpha(s)} = \frac{\sum_{r \in \rho} \alpha(r) P(r, s')}{\alpha(s')} \end{aligned}$$

So far, we did not use the assumption $\Pi = APA$. We now show \circledast and note that we only need $\Pi = APA$ for the direction \iff :

- \implies : this is immediately clear since $\Pi(\rho, \sigma)$ does not depend on s .
- \iff : if we have

$$\frac{\sum_{r \in \rho} \alpha(r) P(r, s)}{\alpha(s)} = \frac{\sum_{r \in \rho} \alpha(r) P(r, s')}{\alpha(s')} \quad \forall s, s' \in \sigma$$

then, it holds that, for any $s \in \sigma$,

$$\frac{\sum_{r \in \rho} \alpha(r) P(r, s)}{\alpha(s)} = \sum_{s' \in \sigma} \alpha(s') \cdot \frac{\sum_{r \in \rho} \alpha(r) P(r, s')}{\alpha(s')} = \sum_{r \in \rho} \alpha(r) \sum_{s' \in \sigma} P(r, s') = (APA)(\rho, \sigma) = \Pi(\rho, \sigma)$$

where the first equality holds since the sum on the right hand side consists of a weighted sum of terms which are equal, and since the weights sum up to 1.

Note that the above proof also works for the continuous-time case. \square

Proposition 13 also allows us to justify the choice $\Pi = AP\Lambda$ (or $\Theta = AQA$) for weighted state space partitioning:

Corollary 14. *Given an irreducible DTMC, assume that the partition Ω is fixed, and that the distributions α are fixed as well (but arbitrary). Further assume that there exists some (arbitrary) matrix $\Pi \in \mathbb{R}^{m \times m}$ such that $\|\Pi A - AP\|_\infty = 0$. Then, it actually holds that $\Pi = AP\Lambda$, i.e. $\Pi = AP\Lambda$ is the unique minimizer of $\|\Pi A - AP\|_\infty$ if the minimum is 0.*

Analogously, for an irreducible CTMC, fixed partition Ω and fixed distributions α , we have the following: if there exists some (arbitrary) matrix $\Theta \in \mathbb{R}^{m \times m}$ such that $\|\Theta A - AQA\|_\infty = 0$, then it actually holds that $\Theta = AQA$.

Proof. This actually already follows from what we have proved in **Proposition 13**: if there is some Π such that $\|\Pi A - AP\|_\infty = 0$, then, by **Proposition 13**, it follows that

$$\forall s \in S : \alpha(s) > 0 \text{ and} \tag{6}$$

$$\forall s, s' \in S \text{ s.t. } \omega(s) = \omega(s') : \forall \rho \in \Omega : \frac{1}{\alpha(s)} \sum_{r \in \rho} \alpha(r)P(r, s) = \frac{1}{\alpha(s')} \sum_{r \in \rho} \alpha(r)P(r, s')$$

Here, we used that the \implies direction in **Proposition 13** does not need the assumption $\Pi = AP\Lambda$ but holds for general Π . But in fact, if we take a closer look at (5) in the proof of **Proposition 13**, we see that furthermore,

$$\|\Pi A - AP\|_\infty = 0 \implies \forall \rho, \sigma \in \Omega : \forall s \in \sigma : \Pi(\rho, \sigma) = \frac{1}{\alpha(s)} \sum_{r \in \rho} \alpha(r)P(r, s)$$

As $\alpha(s) > 0$ by (6), Π is therefore uniquely determined if $\|\Pi A - AP\|_\infty = 0$. We now apply the reverse direction \impliedby of **Proposition 13** with an a priori potentially different matrix Π , namely $\Pi = AP\Lambda$. **Proposition 13** tells us that if (6) holds and if we choose $\Pi = AP\Lambda$, then $\|\Pi A - AP\|_\infty = 0$ which concludes the proof by showing that our unique Π is indeed the matrix $AP\Lambda$. Note again that this proof can also be applied to CTMCs. \square

With the help of **Proposition 13**, we can now apply other results from [10] (we still assume that we aggregate via weighted state space partitioning and that $\Pi = AP\Lambda$, respectively $\Theta = AQA$):

- By [10, Theorem 7], if $\|\Pi A - AP\|_\infty = 0$ and if the DTMC is aperiodic and irreducible, then we have $\|\pi_k^T - p_k^T A\|_1 \rightarrow 0$ for $k \rightarrow \infty$ (i.e. the approximate aggregate probabilities $\pi_k(\sigma)$ converge to the exact aggregate probability $\mathbb{P}[X_k \in \sigma]$) and $\frac{p_k(s)}{\pi_k(\omega(s))} \rightarrow \alpha(s)$. For periodic chains, we have to consider $\frac{1}{k} \sum_{i=1}^k p_i(s)$ instead of $p_k(s)$, see [10, Theorem 6]. In addition, the stationary distribution p of the irreducible DTMC satisfies $p^T = \pi^T A$ where π is the stationary distribution of the aggregated chain in both periodic and aperiodic cases by [10, Theorem 6], which also follows from **Corollary 10**.
- By [10, Theorem 11], if $\|\Theta A - AQA\|_\infty = 0$ and if the CTMC is irreducible, then $\|\pi_t^T - p_t^T A\|_1 \rightarrow 0$ as $t \rightarrow \infty$, and $\frac{p_t(s)}{\pi_t(\omega(s))} \rightarrow \alpha(s)$. Moreover, $p^T = \pi^T A$ holds for the stationary distribution p of the CTMC and the stationary distribution π of the aggregated chain, which again also follows from **Corollary 10**.

The notion of dynamic-exact and exact aggregation has thus already appeared in the literature, but only in the setting of weighted state space partitioning, and [10] is one of the few papers considering this concept. Most papers consider different types of lumpability and aggregatability.

4.2. Dynamic-exactness compared to other lumpability concepts

The following definition was given in [4, Definition 1]:

Definition 15. A partition Ω of the state space of a DTMC is called **ordinarily lumpable** if

$$\forall r, r' \in S \text{ s.t. } \omega(r) = \omega(r') : \forall \sigma \in \Omega : \sum_{s \in \sigma} P(r, s) = \sum_{s \in \sigma} P(r', s) \tag{7}$$

That is, for any two states in the same aggregate, the summed outgoing probabilities to any other aggregate must be identical. For CTMCs, a partition is called ordinarily lumpable if (7) holds with P replaced by Q .

Equivalently, a DTMC is ordinarily lumpable w.r.t. a given partition if there exists a (stochastic) matrix $\Pi \in \mathbb{R}^{m \times m}$ (which is then uniquely defined) such that $\Lambda \Pi = P\Lambda$. For an ordinarily lumpable partition, it is easy to show that $\pi_k(\sigma) = \sum_{s \in \sigma} \tilde{p}_k(s) = \sum_{s \in \sigma} p_k(s)$ if $\Pi = AP\Lambda$, for any $\sigma \in \Omega$, all k , and any initial distribution p_0 (and independent of the choice of α as long as the distributions α are consistent with Ω), i.e. the probability $\mathbb{P}[X_k \in \sigma]$ is exactly equal to $\pi_k(\sigma)$. A simple induction suffices to prove this, see [4, Theorem 5]. The same holds in the continuous-time case.

Note that ordinary lumpability is called strong lumpability in [2], which considers only DTMCs. As before, define the process Y_k by $Y_k = \sigma \in \Omega \iff X_k \in \sigma$. By [2, Theorem 6.3.2], ordinary lumpability is equivalent to the following: for every initial distribution p_0 , Y_k is a Markov chain (whose transition probabilities do not depend on the choice of p_0).

[4, Definition 1] also defines exact lumpability:

Definition 16. A partition $\Omega = \{\Omega_1, \dots, \Omega_m\}$ of the state space of a DTMC is called **exactly lumpable** if

$$\forall s, s' \in S \text{ s.t. } \omega(s) = \omega(s') : \forall \rho \in \Omega : \sum_{r \in \rho} P(r, s) = \sum_{r \in \rho} P(r, s') \tag{8}$$

That is, for any two states in the same aggregate, the summed incoming probabilities from any other aggregate must be identical. For CTMCs, a partition is called exactly lumpable if (8) holds with P replaced by Q .

Furthermore, a partition Ω is called **strictly lumpable** if it is both ordinarily and exactly lumpable.

We can again give an equivalent definition in terms of matrix products: a DTMC is exactly lumpable w.r.t. a given partition if there exists a (not necessarily stochastic) matrix $\tilde{\Pi} \in \mathbb{R}^{m \times m}$ (which is then uniquely defined) such that $\tilde{\Pi}A^\top = A^\top P$. Note that we could also use the following defining equality instead: exact lumpability is equivalent to the existence of a (unique stochastic) matrix $\Pi \in \mathbb{R}^{m \times m}$ such that $\Pi A = AP$ where the rows of A correspond to uniform distributions over the respective aggregates (see Proposition 18). If a partition is ordinarily (exactly) lumpable for a CTMC, then the partition is also ordinarily (exactly) lumpable for any uniformization of the CTMC, regardless of the uniformization rate, see the remarks after [4, Definition 1].

[7, Definition 2.1] also defines lumpability. Note that this definition of lumpability agrees with the definition of ordinary lumpability given above. [7, Definition 2.1] further defines deflatability and aggregatability:

Definition 17. A partition $\Omega = \{\Omega_1, \dots, \Omega_m\}$ of the state space of a DTMC, together with distributions $\alpha_\sigma \in \mathbb{R}^n$ with support on $\sigma \in \Omega$, is called **deflatable** if

$$\forall r \in S : \forall s \in S : P(r, s) = \alpha(s) \cdot \sum_{s' \in \omega(s)} P(r, s') \tag{9}$$

In words: the probability to go from r to s only depends on r and the aggregated state $\omega(s)$ as well as a factor which depends on s , but not on r . Another description: after a jump into a partition element σ , the particular target state can be chosen according to the probability measure α_σ , independently of where the jump started. The partition Ω , together with distributions α , is further called **aggregatable** if it is deflatable and if Ω is ordinarily lumpable.

Definition 17 cannot be extended to CTMCs easily.

Proposition 18. Assume a partition Ω is given for an irreducible DTMC. When setting $\Pi = APA$ and $\alpha(s) = \frac{1}{|\omega(s)|}$, then:

$$\|\Pi A - AP\|_\infty = 0 \iff \Omega \text{ is exactly lumpable}$$

Analogously, when setting $\Theta = AQA$ and $\alpha(s) = \frac{1}{|\omega(s)|}$ for an irreducible CTMC with given partition Ω , we have: $\|\Theta A - AQ\|_\infty = 0$ if, and only if, Ω is exactly lumpable.

Proof. By Proposition 13, $\|\Pi A - AP\|_\infty = 0$ is equivalent to (in the following, we use that $\alpha(s) = \frac{1}{|\omega(s)|}$)

$$\begin{aligned} & \overbrace{\forall s, s' \in S \text{ s.t. } \omega(s) = \omega(s') : \forall \rho \in \Omega :}^{\forall \dots} \alpha(s') \sum_{r \in \rho} \alpha(r) P(r, s) = \alpha(s) \sum_{r \in \rho} \alpha(r) P(r, s') \\ & \iff \forall \dots : \frac{1}{|\omega(s')|} \sum_{r \in \rho} \frac{1}{|\rho|} P(r, s) = \frac{1}{|\omega(s)|} \sum_{r \in \rho} \frac{1}{|\rho|} P(r, s') \\ & \iff \forall \dots : \sum_{r \in \rho} P(r, s) = \sum_{r \in \rho} P(r, s') \\ & \iff \Omega \text{ is exactly lumpable} \end{aligned}$$

The same calculation holds for CTMCs. \square

Proposition 19. Consider an irreducible DTMC, a partition Ω of its state space and distributions $\alpha_\sigma \in \mathbb{R}^n$ with support on $\sigma \in \Omega$. If Ω and the distributions α are deflatable, then $\|\Pi A - AP\|_\infty = 0$.

Proof. Again by Proposition 13, $\|\Pi A - AP\|_\infty = 0$ is equivalent to

$$\begin{aligned} & \overbrace{\forall s, s' \in S \text{ s.t. } \omega(s) = \omega(s') : \forall \rho \in \Omega :}^{\forall \dots} \alpha(s') \sum_{r \in \rho} \alpha(r) P(r, s) = \alpha(s) \sum_{r \in \rho} \alpha(r) P(r, s') \\ \stackrel{\text{deflatability}}{\iff} & \forall \dots : \alpha(s') \sum_{r \in \rho} \alpha(r) \alpha(s) \sum_{\hat{s} \in \omega(s)} P(r, \hat{s}) = \alpha(s) \sum_{r \in \rho} \alpha(r) \alpha(s') \sum_{\hat{s} \in \omega(s')} P(r, \hat{s}) \\ & \iff \forall \dots : \alpha(s') \alpha(s) \sum_{r \in \rho} \alpha(r) \sum_{\hat{s} \in \omega(s)} P(r, \hat{s}) = \alpha(s) \alpha(s') \sum_{r \in \rho} \alpha(r) \sum_{\hat{s} \in \omega(s')} P(r, \hat{s}) \end{aligned}$$

The statement in the last line is true since $\omega(s) = \omega(s')$ by the assumption on s and s' . \square

We next show that none of the lumpability concepts above are necessary conditions for $\| \Pi A - AP \|_\infty = 0$. Hence, except for [10], a large part of the literature has treated stricter than necessary conditions in order for dynamic-exact aggregation to be possible. None of the definitions of ordinary and exact lumpability, as well as deflatability, take into account the initial distribution, so none of these conditions are sufficient for an exact aggregation. Exact lumpability and deflatability only imply dynamic-exactness. At the same time, all three concepts (ordinary & exact lumpability, deflatability) are still useful since they are easier to check computationally, and can thus be relevant for practical applications. This will be discussed in more detail in the next section.

Proposition 20. *There are partitions Ω of the state space of a DTMC and probability distributions α_σ with support on $\sigma \in \Omega$ which are dynamic-exact (when $\Pi = APA$), but where Ω is neither ordinary lumpable, nor exactly lumpable, nor are Ω and the distributions α deflatable.*

Proof. We consider the state space $S = \{1, 2, 3\}$, the aggregation $\Omega = \{\{1\}, \{2, 3\}\}$ and $\alpha(1) = 1, \alpha(2) = \frac{1}{4}, \alpha(3) = \frac{3}{4}$ as well as the DTMC given by the following transition matrix:

$$P = \begin{pmatrix} 0 & \frac{1}{4} & \frac{3}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{4}{9} & \frac{1}{18} & \frac{1}{2} \end{pmatrix} \implies \Pi = \begin{pmatrix} 0 & 1 \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{3}{4} \end{pmatrix}, \quad \Lambda = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

It is easy to check that $\Pi A = AP$ holds, i.e. this aggregation is dynamic-exact and $\| \Pi A - AP \|_\infty = 0$. Indeed, we have

$$\Pi A = \begin{pmatrix} 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{2} \end{pmatrix} = AP$$

We now show that none of the stated properties hold for Ω and α :

- ordinary lumpability: since $\omega(2) = \omega(3)$, by (7) in Definition 15, we would need

$$\underbrace{P(2, 2) + P(2, 3)}_{=1} = \sum_{s \in \{2,3\}} P(2, s) = \sum_{s \in \{2,3\}} P(3, s) = \underbrace{P(3, 2) + P(3, 3)}_{=\frac{5}{9}}$$

which is clearly not true.

- exact lumpability: since $\omega(2) = \omega(3)$, by (8) in Definition 16, we would need $\frac{5}{9} = P(2, 2) + P(3, 2) = P(2, 3) + P(3, 3) = 1$ which is clearly not true.
- deflatability: since $\omega(2) = \omega(3)$, by (9) in Definition 17, we would need

$$\frac{1}{2} = P(2, 2) = \alpha(2) \sum_{s \in \{2,3\}} P(2, s) = \alpha(2) \cdot 1 \implies \alpha(2) = \frac{1}{2}$$

$$\frac{1}{18} = P(3, 2) = \alpha(2) \sum_{s \in \{2,3\}} P(3, s) = \alpha(2) \cdot \frac{5}{9} \implies \alpha(2) = \frac{1}{10}$$

so the given Ω and α are not deflatable, and there is even no other choice of α such that Ω and α would be deflatable.

In fact, there is no aggregation Ω with $\Omega \neq \{S\}$ and $\Omega \neq \{\{1\}, \{2\}, \{3\}\}$ (the trivial partitions) which is ordinary or exactly lumpable, or for which deflatable α distributions exist. \square

We have now seen some concepts which imply dynamic-exactness but not vice versa. However, as mentioned already briefly, dynamic-exactness does imply a type of lumpability: weak lumpability.

Definition 21. Consider a DTMC and a partition Ω of its state space. Denote the state of the chain at time k by X_k . Further define the process $Y_k := \omega(X_k)$. The Markov chain is said to be **weakly lumpable** w.r.t. Ω if there exists an initial distribution p_0 for X_k such that the process Y_k is itself a (time-homogeneous) Markov chain on Ω .

Analogously, we say that a CTMC is weakly lumpable w.r.t. Ω if there exists an initial distribution p_0 such that $Y_t := \omega(X_t)$ is a (time-homogeneous) Markov chain.

Proposition 22. *Consider a DTMC, a partition Ω of its state space and distributions α_σ with support on $\sigma \in \Omega$ such that $\| \Pi A - AP \|_\infty = 0$ (where $\Pi = APA$). Then, the Markov chain is weakly lumpable w.r.t. Ω , and any initial distribution which is compatible with the α distributions will result in the process Y_k being a Markov chain.*

Analogously, if an aggregation of a CTMC with partition Ω satisfies $\| \Theta A - A Q \|_\infty = 0$ (where $\Theta = A Q A$), then the Markov chain is weakly lumpable w.r.t. Ω , and all initial distributions compatible with the α distributions result in Y_t being a Markov chain.

Proof. The discrete-time case was already proven in [2, Theorem 6.4.4].

We look at the continuous-time case. Let p_0 such that $p_0^T A A = p_0^T$ (i.e. p_0 is compatible with the α distributions). We claim that Y_t is a Markov chain with generator Θ in this case. To prove this claim, it is sufficient to show that

$$\underbrace{p_0^T e^{Q t}}_{\text{dist. of } Y_t} A = \underbrace{p_0^T A}_{\text{dist. of } Y_0} \cdot e^{\Theta t} \tag{10}$$

since the above equation shows that the distribution of Y_t can be calculated by applying the generator Θ to the initial distribution of Y_0 . If this holds for arbitrary t , then we can conclude that Y_t is indeed a Markov chain with generator Θ . We proceed to prove that (10) does indeed hold. Note that we have that $AQ^k = AQQ^{k-1} = \Theta AQQ^{k-2} = \dots = \Theta^k A$ by assumption since $\|\Theta A - AQ\|_\infty = 0$.

$$p_0^\top e^{Qt} \Lambda = p_0^\top \Lambda A \left(\sum_{k \geq 0} \frac{t^k Q^k}{k!} \right) \Lambda = \sum_{k \geq 0} \frac{t^k}{k!} \cdot p_0^\top \Lambda A Q^k \Lambda = \sum_{k \geq 0} \frac{t^k}{k!} \cdot p_0^\top \Lambda \Theta^k A \Lambda = p_0^\top \Lambda \left(\sum_{k \geq 0} \frac{t^k \Theta^k}{k!} \right) = p_0^\top \Lambda e^{\Theta t}$$

where we used that $AA = I$. \square

Proposition 23. *There exists a DTMC which is weakly lumpable w.r.t. a partition Ω , but no stochastic matrix Π (not necessarily $\Pi = APA$) and distributions α_σ with support on $\sigma \in \Omega$ exist such that $\|\Pi A - AP\|_\infty = 0$.*

Proof. Consider

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}, \quad \Omega = \{\{1, 2\}, \{3\}\}$$

Note that

$$PA = P \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} = \Lambda \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} =: \Lambda \Pi$$

Hence, this DTMC is ordinarily lumpable w.r.t. Ω . By [2, Theorem 6.4.4] (or by Eq. (3) on page 135 of [2]), the DTMC is therefore also weakly lumpable w.r.t. Ω . In order for $\|\Pi A - AP\|_\infty = 0$ to hold (for some matrix A consistent with Ω and for some matrix Π , not necessarily the Π given above), we would need

$$\begin{aligned} \Pi A &= \begin{pmatrix} \Pi(1, 1) & 1 - \Pi(1, 1) \\ \Pi(2, 1) & 1 - \Pi(2, 1) \end{pmatrix} \begin{pmatrix} \alpha(1) & \alpha(2) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \Pi(1, 1)\alpha(1) & \Pi(1, 1)\alpha(2) & 1 - \Pi(1, 1) \\ \Pi(2, 1)\alpha(1) & \Pi(2, 1)\alpha(2) & 1 - \Pi(2, 1) \end{pmatrix} \\ & \stackrel{!}{=} \begin{pmatrix} \frac{1}{4}\alpha(2) & \frac{1}{2}\alpha(1) + \frac{1}{4}\alpha(2) & \frac{1}{2}\alpha(1) + \frac{1}{2}\alpha(2) \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} = AP \end{aligned}$$

Looking at the lower right entry, we find that $\Pi(2, 1) = \frac{3}{4}$ would need to hold, and looking at the lower left entry, we then see that $\alpha(1) = \frac{2}{3}$ and hence $\alpha(2) = \frac{1}{3}$. But then, the upper right entry of AP would be $\frac{1}{2}\alpha(1) + \frac{1}{2}\alpha(2) = \frac{1}{2}$, implying that we would need $\Pi(1, 1) = \frac{1}{2}$, which yields

$$\Pi A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \neq \begin{pmatrix} \frac{1}{12} & \frac{5}{12} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} = AP$$

Thus, it is indeed impossible in this example to choose a stochastic matrix Π and a matrix A with α distributions as rows leading to $\|\Pi A - AP\|_\infty = 0$ for the weakly lumpable $\Omega = \{\{1, 2\}, \{3\}\}$. As Ω is also ordinarily lumpable, we further see that ordinary lumpability does not imply dynamic-exactness, either. \square

4.3. Choosing the aggregated transition matrix when using weighted state space partitioning

In the current Section 4, we mentioned repeatedly that we choose $\Pi = APA$ (respectively $\Theta = AQA$) when we aggregate a Markov chain using weighted state space partitioning, which was justified by Corollary 14. We now want to take a closer look at whether this is indeed the best choice, or if we can achieve a lower error bound by using a different matrix for Π or Θ . In the following, we will focus on the discrete time case, but as we only consider the problem of finding a matrix Π such that $\|\Pi A - AP\|_\infty$ is minimal (when A and P are fixed), the calculations below also hold for CTMCs.

We start by introducing the median-based scheme to determine Π . This scheme might achieve lower error bounds than setting $\Pi = APA$ and can be applied in the context of weighted state space partitioning. [5] already introduced this method for the special case where the distributions α_σ are uniform distributions on the aggregates. The median-based scheme works as follows: assume that the partition Ω and the distributions α_σ are fixed (hence A and Λ are fixed, we now want to determine Π). We then set $\Pi(\rho, \sigma)$ to the weighted median of the (multi-)set with elements

$$a_s = \frac{1}{\alpha(s)} \sum_{r \in \rho} \alpha(r) P(r, s) \quad (s \in \sigma \text{ with } \alpha(s) > 0) \tag{11}$$

weighted with the weights $\alpha(s)$ for every state $s \in \sigma$. The weighted median of the (multi-)set $\{a_s \mid s \in \sigma \text{ with } \alpha(s) > 0\}$ is the element $a_{s_{\text{med}}}$ with the property that

$$\sum_{\substack{s \in \sigma : \alpha(s) > 0 \\ a_s < a_{s_{\text{med}}}}} \alpha(s) \leq \frac{1}{2} \quad \text{and} \quad \sum_{\substack{s \in \sigma : \alpha(s) > 0 \\ a_s > a_{s_{\text{med}}}}} \alpha(s) \leq \frac{1}{2} \tag{12}$$

It may happen that there are multiple elements in the (multi-)set $\{a_s \mid s \in \sigma \text{ with } \alpha(s) > 0\}$ which satisfy (12). In such a case, the weighted median is non-unique, and any of the elements satisfying (12) may be chosen. For a fixed partition Ω and fixed α_σ distributions, setting $\Pi(\rho, \sigma)$ to this weighted median minimizes the error bound $\|\Pi A - AP\|_\infty$ as we will see in Proposition 24.

Remark. Using the median-based scheme can result in matrices Π and Θ which are no longer stochastic or a generator. Therefore, the aggregated transient distributions π_k and π_i will also no longer necessarily be probability distributions in this case.

Proposition 24. *Given a DTMC, assume that the partition Ω is fixed, and that the distributions α are fixed as well (but arbitrary). Then, setting Π according to (11) and (12) minimizes $(|\Pi A - AP| \cdot \mathbf{1}_n)(\rho)$ for every $\rho \in \Omega$ among all matrices $\Pi \in \mathbb{R}^{m \times m}$. In particular, $\|\Pi A - AP\|_\infty$ is minimized.*

Analogously, for a CTMC, fixed partition Ω and fixed distributions α , setting Θ according to (11) and (12) minimizes $(|\Theta A - AQ| \cdot \mathbf{1}_n)(\rho)$ for every ρ .

Proof. We write out the definition of $(|\Pi A - AP| \cdot \mathbf{1}_n)(\rho) (= \tau(\rho))$ in [5], similarly to what we saw in (5):

$$\begin{aligned} (|\Pi A - AP| \cdot \mathbf{1}_n)(\rho) &= \sum_{s \in S} |(\Pi A)(\rho, s) - (AP)(\rho, s)| = \sum_{\sigma \in \Omega} \sum_{s \in \sigma} \left| \alpha(s) \Pi(\rho, \sigma) - \sum_{r \in \rho} \alpha(r) P(r, s) \right| \\ &= \sum_{\sigma \in \Omega} \sum_{s \in \sigma} \underbrace{\alpha(s) \cdot \left| \Pi(\rho, \sigma) - \frac{1}{\alpha(s)} \sum_{r \in \rho} \alpha(r) P(r, s) \right|}_{=: \tau(\rho, \sigma)} \end{aligned}$$

Minimizing $(|\Pi A - AP| \cdot \mathbf{1}_n)(\rho)$ (with α and Ω fixed) amounts to minimizing $\tau(\rho, \sigma)$ separately for each $\sigma \in \Omega$, since we can choose a different value for $\Pi(\rho, \sigma)$ for every σ . We thus want to set $\Pi(\rho, \sigma)$ by solving the following optimization problem:

$$\Pi(\rho, \sigma) := \arg \min_{x \in \mathbb{R}} \sum_{s \in \sigma} \alpha(s) \cdot \left| x - \underbrace{\frac{1}{\alpha(s)} \sum_{r \in \rho} \alpha(r) P(r, s)}_{a_s} \right|$$

Note: it is easy to see that we can assume $\alpha(s) > 0$ for all $s \in S$ w.l.o.g. in this proof. We show that the weighted median of the (multi-)set $\{a_s \mid s \in \sigma\}$ with weights $\alpha(s)$, $s \in \sigma$ does indeed solve the given minimization problem. For ease of notation and w.l.o.g., we write $\{a_s \mid s \in \sigma\} = \{a_1, a_2, \dots, a_k\}$, we assume $a_1 < a_2 < \dots < a_k$ (we can merge elements a_i and a_{i+1} if $a_i = a_{i+1}$ into a new element whose weight is the sum of the two weights) and we denote the corresponding weights by $\alpha(1), \dots, \alpha(k)$. Consider the function $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = \sum_{i=1}^k \alpha(i) |x - a_i|$. Note that this function is piecewise linear: for $x \in (a_i, a_{i+1})$, f is a linear combination of linear functions and thus linear. On the other hand, when $x = a_i$ for some i , the slope of f might change.

By this piecewise linearity and since $\lim_{x \rightarrow \pm\infty} f(x) = \infty$, we know that f must take a global minimum for some x in the set $\{a_1, \dots, a_k\}$. Denote the value of x for which the minimum is taken by a_{i^*} . Assume for a contradiction that a_{i^*} is not the weighted median. W.l.o.g. we consider the case where $\sum_{i>i^*} \alpha(i) > \frac{1}{2}$, i.e. the case where the total weight of the elements which are bigger than a_{i^*} is bigger than $\frac{1}{2}$. Note that we also assumed above (w.l.o.g.) that no two elements in the set $\{a_s \mid s \in \sigma\}$ are identical, i.e. we have $a_{i^*+1} > a_{i^*}$.

We proceed to show that $f(a_{i^*+1}) < f(a_{i^*})$, and hence the minimum is not taken at a_{i^*} and our assumption that a_{i^*} is not the weighted median must have been wrong. To see that this is true, consider the following:

$$\begin{aligned} f(a_{i^*+1}) &= \sum_{i=1}^k \alpha(i) |a_{i^*+1} - a_i| \\ &= \sum_{i=1}^{i^*} \alpha(i) |a_{i^*+1} - a_i| + \underbrace{\alpha(i^*+1) |a_{i^*+1} - a_{i^*+1}|}_0 + \sum_{i=i^*+2}^k \alpha(i) |a_{i^*+1} - a_i| \\ &= \sum_{i=1}^{i^*} \alpha(i) (|a_{i^*} - a_i| + |a_{i^*+1} - a_{i^*}|) + \sum_{i=i^*+2}^k \alpha(i) (|a_{i^*} - a_i| - |a_{i^*+1} - a_{i^*}|) \\ &= \underbrace{\sum_{i=1}^k \alpha(i) |a_{i^*} - a_i|}_{f(a_{i^*})} + |a_{i^*+1} - a_{i^*}| \underbrace{\left(\underbrace{\sum_{i=1}^{i^*} \alpha(i)}_{< \frac{1}{2}} - \underbrace{\sum_{i=i^*+1}^k \alpha(i)}_{> \frac{1}{2}} \right)}_{< 0} \end{aligned}$$

Note that the additional term (if compared to the previous line) in the leftmost sum on the last line is compensated by the additional term in the rightmost sum.

The proof above also works for the continuous-time case. \square

So we can achieve a lower error bound by using the median-based scheme instead of $\Pi = APA$, at the cost of Π not necessarily being stochastic anymore. Therefore, $\Pi = APA$ could still be considered as the natural choice for Π , because this guarantees stochasticity, and as this choice is optimal when we can achieve an error bound of 0, at least. As a final side remark, note that Proposition 13 implies that all elements in the multiset a_s from Eq. (11) are equal when there is some Π such that $\|\Pi A - AP\|_\infty = 0$.

4.4. Comparison with more general state space reduction schemes

Consider again the example

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

which was already given in the proof of Proposition 23. There, it was already shown that $\Omega = \{\{1, 2\}, \{3\}\}$ is not a dynamic-exact aggregation when using weighted state space partitioning. With similar calculations, we can see that $\{\{1, 3\}, \{2\}\}$ and $\{\{1\}, \{2, 3\}\}$ are no dynamic-exact aggregations, either. Hence, it is impossible to reduce the state space of this Markov chain to two states with a dynamic-exact aggregation under weighted state space partitioning. However, we actually have

$$A = \begin{pmatrix} \frac{7}{\sqrt{213}} & \frac{8}{\sqrt{213}} & \frac{10}{\sqrt{213}} \\ \frac{44}{\sqrt{3621}} & -\frac{41}{\sqrt{3621}} & \frac{2}{\sqrt{3621}} \end{pmatrix} \approx \begin{pmatrix} 0.48 & 0.55 & 0.69 \\ 0.73 & -0.68 & 0.03 \end{pmatrix}, \quad \Pi = \begin{pmatrix} 1 & 0 \\ \frac{1}{4\sqrt{17}} & -\frac{1}{4} \end{pmatrix} \approx \begin{pmatrix} 1 & 0 \\ 0.06 & -0.25 \end{pmatrix}$$

$\implies \Pi A = AP$

The matrices above were computed using the Schur decomposition as demonstrated in the proof of Proposition 12. So, we can, in fact, find a reduction to a state space of dimension 2 which is dynamic-exact if we allow general matrices Π and A and do not restrict ourselves to weighted state space partitioning. The more general view of state space reduction actually allows us to find dynamic-exact aggregations for (almost) any desired number of aggregates, as shown in Proposition 12, but the above example demonstrates that this is not true for weighted state space partitioning. We now further fix the initial vector $p_0 = (\frac{19}{30}, 0, \frac{11}{30})^T$. If we use the above dynamic-exact aggregation, we find that the optimal choice for π_0 is (this can be calculated using a linear program):

$$\pi_0 = \left(\frac{779}{90\sqrt{213}}, \frac{152\sqrt{17}}{90\sqrt{213}} \right)^T \approx (0.59, 0.48)^T \text{ resulting in } \pi_0^T A = \left(\frac{19}{30}, 0, \frac{19}{45} \right) \text{ and } \|\pi_0^T A - p_0^T\|_1 = \frac{1}{18} \approx 0.06$$

Note that $\pi_0^T A \geq p_0^T$, and thus, by Proposition 9, $\tilde{p}_k \geq p_k$ for all k , and the error is $\frac{1}{18}$ for every k . If we look instead at the best possible weighted state space partitioning with 2 aggregates, $\Pi = APA$ and with π_0 a probability distribution, we find that the choice minimizing the error bound after one step, i.e. the sum of the initial error and $\|\Pi A - AP\|_\infty$, is the following (which we found by brute force search with a precision of 4 digits after the decimal point):

$$A \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.46 & 0.54 \end{pmatrix}, \quad \Pi \approx \begin{pmatrix} 0 & 1 \\ 0.38 & 0.62 \end{pmatrix}, \quad \pi_0 = \left(\frac{19}{30}, \frac{11}{30} \right)^T$$

$\implies \|\Pi A - AP\|_\infty \approx 0.07, \quad \|\pi_0^T A - p_0^T\|_1 \approx 0.34$

$\tilde{p}_1 \approx (0.14, 0.40, 0.46)^T, \quad \tilde{p}_{100} \approx (0.28, 0.34, 0.39)^T$

$p_1 \approx (0.18, 0.41, 0.41)^T, \quad p_{100} \approx (0.28, 0.32, 0.4)^T = \text{stationary distribution of } P$

Here, $\|\tilde{p}_1 - p_1\|_1 \approx 0.10$. Therefore, the dynamic-exact aggregation achieved a slightly lower error (of 0.06) in the approximation of p_1 when compared to the best possible weighted state space partitioning. However, we see that, as the transient distributions approach stationarity, the actual error of the latter becomes smaller (around 0.03 at step 100, compared to the constant error 0.06 of the dynamic-exact aggregation), even though the first row of A in the dynamic-exact aggregation is actually a multiple of the stationary distribution of P (and it is thus possible to have $\pi_k^T A$ equal the stationary distribution).

This example shows that it can be useful to consider the more abstract view of aggregation in some cases, achieving a lower approximation error. On the other hand, the advantage of weighted state space partitioning is that we obtain an intuitive reduction to a Markov chain with fewer states which we can interpret in a meaningful way, while the more abstract view only results in an abstract lower-dimensional state space and the time evolution on this reduced state space is represented by a general linear map, and not necessarily by a Markov chain.

4.5. Summary of results

In Fig. 1, the relation between the different notions of lumpability is shown with the help of a Venn diagram. Deflatability is only defined for DTMCs, which is why it is depicted with a dashed circle. For the other properties, the diagram is true for both the discrete- and the continuous-time case. The whole diagram should be understood as comparing properties of state space partitions, and we look at dynamic-exactness in the restricted context of weighted state space partitioning here. The inclusion of the exactly lumpable and deflatable sets in the dynamic-exact set was proven in Propositions 18 and 19, and the inclusion of the dynamic-exact

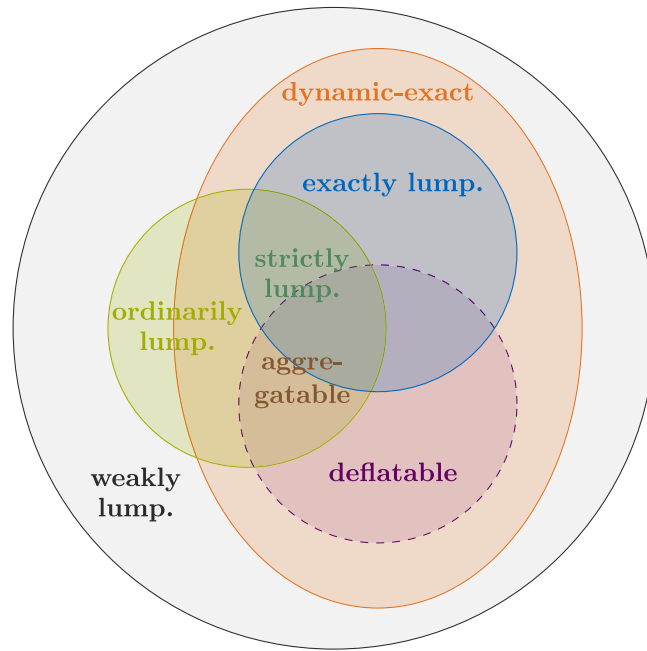


Fig. 1. A Venn diagram summarizing the relation between the different types of lumpability.

set in the weakly lumpable set was show in Proposition 22. On the other hand, Propositions 20 and 23 demonstrated that these inclusions are real inclusions and not equalities. The example in the proof of Proposition 23 also showed that the ordinarily lumpable set is not included in the dynamic-exact set. The fact that all the other depicted non-empty intersections of the various properties exist is left for the reader to check, but very simple examples do usually suffice (e.g. to see that exactly lumpable and deflatable partitions exist which are not ordinarily lumpable).

5. Finding suitable aggregations

We will now consider how we can, in practice, find suitable aggregations which achieve a low error bound. In contrast to the Schur decomposition approach, which was already presented in Proposition 12, the algorithms below will all use the weighted state space partition approach and we will mostly stick to the notation used in the previous Section 4. The goal of these algorithms is to choose a partition Ω of the state space S of a DTMC or CTMC in such a way that the error bound $\|IIA - AP\|_\infty$ (and potentially also the initial error) is small, as this will result in a good approximation \tilde{p}_k of the transient distribution p_k . At the same time, we would like $|\Omega| = m \ll n = |S|$ in order to reduce the computational effort required to calculate \tilde{p}_k . An ideal algorithm would receive a parameter ϵ as input and determine the partition Ω with the fewest aggregates satisfying $\|\pi_0^\top A - p_0^\top\|_1 + k \cdot \|IIA - AP\|_\infty < \epsilon$, where k is the time point for which we would like to compute an approximation of the transient distribution. This is motivated by Theorems 4 and 5 and would guarantee an error of at most ϵ .

Solving the above problem exactly will in general result in a runtime exceeding the time needed to simply compute p_k exactly for the original chain. We will therefore consider different ways of choosing an Ω which is somehow close to the optimal solution. The presented algorithms can be used for general irreducible Markov chains, without any assumption on the structure of the chain. However, they can only be expected to perform well if a certain structure is present: Sections 3.3 and 4 characterize settings in which the error bounds are low, and which can be identified with comparatively little computational effort, in contrast to finding a partition which satisfies exactness as defined in Definition 8. The algorithms presented in this section will thus try to identify aggregates which are close to fulfilling conditions such as exact lumpability or aggregatability. We will start with some remarks on the time needed to calculate the exact transient distribution of a Markov chain, in order to be able to assess the speed-up resulting from aggregation.

5.1. No aggregation

If we perform no aggregation and compute exact transient distributions, then we get the following runtimes:

- For DTMCs, computing p_k amounts to k vector-matrix multiplications (each of the form $p_{i+1}^\top = p_i^\top P$) of a vector of length n with a matrix of size $n \times n$. Each such multiplication has a runtime of order $\mathcal{O}(n^2)$, and since we need k of those, the total runtime amounts to $\mathcal{O}(kn^2)$. Alternatively, we can compute P^k and then multiply with p_0 . This will need around $\mathcal{O}(n^3 \log(k))$ time (or $\mathcal{O}(M(n) \log(k))$ where $M(n)$ is the time needed to multiply two square matrices of size n).

- For CTMCs, computing the exact transient distribution p_t at time t is not feasible in general. Instead, using uniformization and truncation (see [5, Section 2.5] and [15]), the transient distribution can be computed up to a pre-defined error ϵ (we will neglect the dependence of the runtime on ϵ , which we assume to be fixed). The runtime of this computation is $\mathcal{O}(qtn^2)$ where $q := \max_{s \in S} |Q(s, s)|$.

5.2. Almost aggregatability

In Definition 17, aggregatability for DTMCs was defined, and we have seen in Proposition 19 that deflatability actually already implies that the aggregation is dynamic-exact and hence $\|\Pi A - A P\|_\infty = 0$. The first algorithm, based on [7], will thus try to identify a partition which is close to an aggregatable partition (or “almost aggregatable”, see [7, Definition 2.8]). As the definition of deflatability only applies to DTMCs, this partitioning algorithm only works for DTMCs a priori.

We quickly summarize the algorithm from [7] here for better understanding of some of the choices available when implementing it. Note that in [7], the transpose of the transition matrix P is considered, and all transient distributions are column vectors instead of row vectors. In the following summary, we will stick to our notation, and hence present a transposed version of the results in [7].

We first consider the case of an aggregatable matrix P , which means that $\Lambda \Pi A = P$ by [7, Proposition 2.6], where the partition corresponding to Λ together with the α distributions contained in A is aggregatable and where $\Pi = A P A$. We will write $\tilde{P} := \Lambda \Pi A$ (so we now consider the case where $\tilde{P} = P$). The idea is to find a connection between the singular value decompositions (SVD for short, also see [11, Section 2.4]) of Π and P which allows for identification of the aggregates by analyzing the singular value decomposition of P , without knowledge of Π or Ω . For an almost aggregatable P , the changes in the singular value decomposition compared to the closest aggregatable matrix will be small, and the algorithm can thus also be applied if a matrix is only almost aggregatable. For a detailed justification, we refer to [7].

In order to understand the connection between the singular value decomposition and Ω , we go back to assuming that P is aggregatable and that Ω and the α distributions are given. Consider matrices

$$D_A = \begin{pmatrix} \|\alpha_{\Omega_1}\|_2 & & \\ & \ddots & \\ & & \|\alpha_{\Omega_m}\|_2 \end{pmatrix} \quad D_A = \begin{pmatrix} \sqrt{|\Omega_1|} & & \\ & \ddots & \\ & & \sqrt{|\Omega_m|} \end{pmatrix}$$

Let $\hat{U} \hat{\Sigma} \hat{V}$ be the singular value decomposition of $D_A \Pi D_A$ (note: we do *not* transpose \hat{V}), i.e. $\hat{U} \hat{\Sigma} \hat{V} = D_A \Pi D_A$, \hat{U} and \hat{V} are orthogonal (their rows and columns are orthonormal vectors), and $\hat{\Sigma}$ contains the singular values on its diagonal, ordered from largest to smallest value. Then, the singular value decomposition $U \Sigma V$ of P can be written as

$$U = \left(\Lambda D_A^{-1} \hat{U} \mid U^{(2)} \right) \in \mathbb{R}^{n \times n} \quad \Sigma = \begin{pmatrix} \hat{\Sigma} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n} \quad V = \begin{pmatrix} \hat{V} D_A^{-1} A \\ V^{(2)} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

where $U^{(2)}$ and $V^{(2)}$ are such that U and V are orthogonal. This is verified by computing:

$$U \Sigma V = \Lambda \overbrace{D_A^{-1} \hat{U} \hat{\Sigma} \hat{V} D_A^{-1}}^{\Pi} A = \tilde{P} = P$$

Details can be found in [7], where orthogonality of U and V is shown as well. Note that we used the matrices D_A and D_A only in order to scale the singular values of Π such that they agree with the singular values of P , they have no other relevance. For simplicity, we will assume that the singular value decomposition of P is unique if the singular values in Σ are ordered by size, and if we ignore $U^{(2)}$ and $V^{(2)}$ which are anyway not used anymore. This is true, for example, when Π has full rank (i.e. rank m) and all non-zero singular values of P have multiplicity one (see [16, Lecture 4 and Theorem 4.1]).

We now consider the first m rows of V (which are the first m right-singular vectors of P), i.e. $\hat{V} D_A^{-1} A \in \mathbb{R}^{m \times n}$. We call the i th column of this submatrix $v(i) = (V(1, i), \dots, V(m, i))^T \in \mathbb{R}^m$. Note that i corresponds to state i of the Markov chain, and hence $\omega(i)$ is the corresponding aggregate. Using that the upper m rows of V are equal to $\hat{V} D_A^{-1} A$, we can write

$$v(i) = \|\alpha_{\omega(i)}\|_2^{-1} \alpha(i) \cdot \underbrace{\left(\hat{V}(1, \omega(i)), \dots, \hat{V}(m, \omega(i)) \right)^T}_{\omega(i)\text{th column of } \hat{V}} \in \mathbb{R}^m$$

This implies that for two states r, s in the same aggregate, $v(r)$ and $v(s)$ will point in the same direction — in fact,

$$\forall r, s \in S \text{ s.t. } \omega(r) = \omega(s) : \quad \mathbb{R}^m \ni v(r) = \underbrace{\frac{\alpha(r)}{\alpha(s)}}_{\in \mathbb{R}} \cdot v(s) \in \mathbb{R}^m \tag{13}$$

Furthermore, by orthogonality of the columns of \hat{V} , $v(r)$ and $v(s)$ will be orthogonal if $\omega(r) \neq \omega(s)$. This structure of the submatrix can be exploited to recover Ω (and even α). For almost aggregatable P , we can only expect (13) to hold approximately. The number of aggregates m is furthermore unknown. Instead, we choose an integer l (details on how to choose l follow later) and then consider the first l rows of V (instead of the first m rows). This leads to the following three possible algorithms to compute Ω if l is already fixed:

- **SVDsgn**: Proposed as a very simple algorithm in [7] with only limited practical applicability due to its numerical instability. The aggregates are recovered by putting two states r and s into the same aggregate if the sign structure of the vectors $v(r)$ and $v(s)$ is identical. By (13) and by orthogonality of the vectors v for states in different aggregates, this yields the correct partition Ω if P is aggregatable. However, for almost aggregatable P , perturbed values in the vectors v can lead to the sign of an entry changing, resulting in instability.
- **SVDseba**: Proposed as a more stable algorithm in [7] via a combination with [17]. The sparse eigenbasis approximation algorithm proposed in [17] is applied to the first l rows of V . This results in an approximate sparse basis of the space spanned by the first l rows of V . In the case of aggregatability of P , the space spanned by the first m rows of V is spanned by the vectors $\alpha_{\Omega_1}, \dots, \alpha_{\Omega_m}$ as a consequence of (13). Therefore, the sparse basis obtained from applying the algorithm of [17] should approximately correspond to the α distributions for almost aggregatable P .
- **SVDdir**: A new proposal presented in this paper, with the intention to fully exploit (13). States are clustered such that the distance between the corresponding v vectors within a cluster should be low, where the distance is measured as follows: the shorter vector is projected onto the longer vector, and we then measure the euclidean distance between the shorter original and the projected vector (see Algorithm 1 below). The intention behind this is to measure whether two vectors $v(r)$ and $v(s)$ point in approximately the same direction, as should be the case for r and s in the same aggregate by (13). The distance between vectors is not measured as the angle between vectors because this approach would suffer from the same numerical instability as SVDsgn: if the entries of a v vector are close to 0, small perturbations can lead to huge changes in the angle of the vector. An additional ordering of the v vectors by length is applied to increase stability. We give the implementation details in Section 5.2.1.

With a given l corresponding to the actual number of aggregates, the algorithms above can recover Ω from the singular value decomposition of an almost aggregatable matrix P . However, the number of aggregates is in general not known in advance. In order to choose l , it makes sense to analyze the spectrum of singular values of P . One way would be to identify a gap in this spectrum and set l to the number of singular values above the gap. We chose a different approach which delivered better results in experiments. Call the singular values $\gamma_1 \geq \dots \geq \gamma_n \geq 0$. Given a threshold parameter ε , l is set to the smallest value such that

$$\sum_{i=1}^l \gamma_i \geq (1 - \varepsilon) \sum_{i=1}^n \gamma_i \tag{14}$$

i.e. the first l singular values sum to at least $1 - \varepsilon$ times the sum of all singular values. If the computation of all singular values is too expensive, an alternative is to simply choose a fixed number l (or try multiple different l) which is small enough such that the computation of the first l singular values and corresponding right-singular vectors is still affordable (there are algorithms which allow one to compute only a part of the singular value decomposition, often referred to as truncated SVD; compare with [11, Section 10.4]).

After having calculated Ω using one of the above SVD algorithms, it is also necessary to specify the corresponding α distributions. This can be done by comparing the length of the vectors $v(s)$ for all states s in an aggregate. However, we saw better results in our experiments with the following simple approach for setting α once the aggregation is fixed:

$$\alpha(s) = \frac{\sum_{r \in S} P(r, s)}{\sum_{r \in S} \sum_{s' \in \omega(s)} P(r, s')} \tag{15}$$

$\alpha(s)$ is the same as the probability of being in state s , conditioned on being in the aggregate of s , after the Markov chain took a single step, starting with a uniform distribution. Intuitively, the distributions α_σ should be approximations of this type of conditional probabilities, with the exception that we do not necessarily start with a uniform distribution. Whenever Ω is fixed and α is chosen according to (15), we will refer to these α distributions as proportional α . Note: proportional α is well-defined if the chain is irreducible. It is easy to see that for a deflatable partition Ω , choosing α as in (15) will result in the corresponding α distributions which make the partition Ω deflatable.

The runtime of the SVD algorithms depends on the variant chosen. For the simple SVDsgn, the computational cost is dominated by the singular value decomposition, which needs $\mathcal{O}(n^3)$ time (see [11, Figure 8.6.1]). We saw that computing p_k without aggregation results in a complexity of $\mathcal{O}(kn^2)$. Asymptotically (for large k and n), applying the SVD algorithm therefore only makes sense if $k \gg n$. As noted in [7], we can reduce the complexity of the SVD approach by random sampling of the entries of P under some assumptions, which makes the algorithm more attractive. See [7] for details. In our experiments, we also used a singular value decomposition algorithm for dense and sparse matrices which only computes a partial singular value decomposition, offered by the SciPy python package.¹ This reduces the runtime from $\mathcal{O}(n^3)$ to approximately $\mathcal{O}(l \cdot z)$ where l is the number of singular values to compute and z is the number of non-zero entries of the matrix P (see [18, Section 1.8]). For dense matrices, we therefore have a reduced runtime of $\mathcal{O}(l \cdot n^2)$. We can thus choose l small enough such that we can afford running the algorithm in practice (and such that we actually achieve a speed-up compared to the exact calculation). This approach comes at the cost of potentially losing precision as only a part of the singular vectors are considered, and we can no longer choose l according to an informed heuristic such as (14). More implementation details are given in the following section and at the beginning of Section 6 for modifications which improve the runtime.

¹ See the SciPy documentation at <https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.svds.html>.

As mentioned before, the three variants of the SVD algorithm can only be applied to DTMCs. They are expected to perform well for almost aggregatable P , and this is confirmed in the experiments. However, in different settings, identifying almost exactly lumpable partitions may provide better results, with the additional benefit that this is also possible for CTMCs. This is discussed in Section 5.3.

5.2.1. Implementation of SVDdir

We briefly summarize the implementation details of SVDdir for completeness. First, we define the function which computes a distance indicating whether two vectors point in approximately the same direction, which should also work if vectors with entries close to zero are perturbed randomly.

Algorithm 1 vdist (measuring whether vectors point in a similar direction)

Input: vectors $v_1, v_2 \in \mathbb{R}^l$ s.t. $\|v_1\|_2 \geq \|v_2\|_2$

Output: a real number ≥ 0 , which roughly measures a distance between vector angles

1: $v_{\text{proj}} \leftarrow$ projection of v_2 onto v_1

2: **return** $\|v_{\text{proj}} - v_2\|_2$

We can now look at the implementation details of SVDdir, given in Algorithm 2. After calculating the singular value decomposition of P and after determining the number of rows of V to consider with the aid of (14), the algorithm iterates over all states $s \in S$ in descending order of the length of $v(s)$ (in line 13 of Algorithm 2). This ordering is used to increase stability: the larger the entries of a vector, the less its angle varies under small perturbations of the vector components. Hence, the first assignments of states to aggregates are based on longer vectors whose angles are most reliable.

The assignment to aggregates proceeds as follows: when processing state s , the minimum distance d_{\min} of $v(s)$ (according to vdist, see Algorithm 1) to the vectors $v(s_1), \dots, v(s_i)$ is calculated, where s_1, \dots, s_i are all states which have already been processed. If this distance is smaller than the parameter δ (i.e. if $d_{\min} < \delta$), then s is assigned to the aggregate of s_{i^*} where the distance $\text{vdist}(v(s), v(s_{i^*}))$ is minimal. In another effort to increase numerical stability, the distance d_{\min} is first only calculated for the states in $S_{\text{reliable}} = \{s' \in \{s_1, \dots, s_i\} : \|v(s')\|_2 > 2\delta\}$, and an assignment to an aggregate of one of the states in S_{reliable} is performed if $d_{\min} < \delta$ on this set, and only if this is not possible, we calculate distances to all previously processed states. This additional step follows the same reasoning as before: the angle of short perturbed vectors is uncertain, so the aggregate assignment is more certain if we assign s to an aggregate which contains a state with a longer vector pointing in a similar direction as $v(s)$. If no previously processed state s' satisfies $\text{vdist}(v(s), v(s')) < \delta$, then s is assigned to a new aggregate.

As input for Algorithm 2, ε should be chosen according to how close the resulting aggregation should be to an aggregatable partition. For $\varepsilon = 0$ (and $\delta = 0$), the algorithm finds aggregatable partitions. For $\varepsilon = 1$, all states are assigned to a single aggregate. We used $\delta = 0.05$ in all our experiments. Using a constant δ regardless of the size of the state space makes sense insofar as that the columns of V are vectors of unit length. However, since we crop the columns to dimension l (which results in shorter vectors in general), a δ depending on l might also be a good choice. We experimented with different δ values, and the results do not seem to be very sensitive to which δ is chosen (aggregations of similar quality are found for different δ values when adapting ε appropriately). An in-depth analysis of the numerical stability of Algorithm 2 is still missing, but we observed at least a convincing performance in our experiments.

We conclude this section with a short runtime analysis of Algorithm 2. Line 2 (the singular value decomposition) takes $\mathcal{O}(n^3)$ time, as stated before (see [11, Figure 8.6.1]). Lines 1 and 3 to 11 are negligible in comparison. Line 12 (sorting the vectors by descending length) takes time $\mathcal{O}(n \log(n))$ after having computed all vector lengths in time $\mathcal{O}(nl)$; both runtimes are smaller than $\mathcal{O}(n^3)$. In line 13, we loop over all n states. Determining S_{reliable} (lines 14 and 15) can be done in time $\mathcal{O}(1)$ by successively adding each processed state with suitable length to S_{reliable} . Line 16 takes $\mathcal{O}(nl)$ time at most (we need to call vdist at most n times, and vdist is applied to vectors of dimension l). The same holds for line 20. The remaining part of the loop body is negligible. We arrive at $\mathcal{O}(nl)$ for the loop body, yielding a total runtime of the loop in lines 13 to 28 of $\mathcal{O}(n^2l)$ which is also smaller than $\mathcal{O}(n^3)$. The overall runtime is thus still dominated by the singular value decomposition with $\mathcal{O}(n^3)$.

For larger state spaces, we only compute partial singular value decompositions (for dense matrices in time $\mathcal{O}(ln^2)$ with l predefined) to circumvent the $\mathcal{O}(n^3)$ runtime, resulting in the state clustering time (the loop from lines 13 to 28) having the same asymptotical runtime of $\mathcal{O}(ln^2)$. Asymptotically, this results in the computation being faster than calculating exact transient distributions for time k if $l \ll k$.

5.3. Almost exact lumpability

By Proposition 18, if Ω is exactly lumpable, if the α distributions are uniform distributions, and if $\Pi = APA$ (or $\Theta = AQA$), then the aggregation is dynamic-exact. However, we cannot expect an exactly lumpable partition to exist for a general Markov chain. It might be, though, that a partition exists which is close to being exactly lumpable. This motivates the following definition:

Algorithm 2 Calculating almost aggregatable partitions with SVDdir

Input: a Markov chain, defined via its transition matrix P on state space $S = \{1, \dots, n\}$,
the parameter ε , and the parameter δ

Output: an aggregation function ω
whose corresponding partition is close to an aggregatable partition

```

1:  $\omega \leftarrow ((s \in S) \mapsto 1)$  ▷ aggregation function
2:  $U, \Sigma, V \leftarrow$  singular value decomposition of  $P$  ▷ s.t.  $P = U\Sigma V$ , not  $P = U\Sigma V^T$ , with ordered singular values in  $\Sigma$ 
   ▷  $\gamma_1 \geq \dots \geq \gamma_n \geq 0$  are the singular values of  $P$ 
   ▷ number of considered rows of  $V$ 
   ▷ determine  $l$  according to (14)
3:  $\gamma_1, \dots, \gamma_n \leftarrow$  values on diagonal of  $\Sigma$ 
4:  $l \leftarrow 0$ 
5: while  $\sum_{i=1}^l \gamma_i < (1 - \varepsilon) \sum_{i=1}^n \gamma_i$  do
6:    $l \leftarrow l + 1$ 
7: end while
8: for all  $s \in \{1, \dots, n\}$  do
9:    $v(s) \leftarrow (V(1, s), \dots, V(l, s))^T \in \mathbb{R}^m$  ▷ extract cropped columns from  $V$ 
10: end for
11:  $m \leftarrow 0$  ▷ current number of aggregates
12:  $(s_1, \dots, s_n) \leftarrow$  permutation of  $(1, \dots, n)$  s.t.
    $\|v(s_1)\|_2 \geq \dots \geq \|v(s_n)\|_2$  ▷ go through  $v(s_i)$  by descending length
13: for all  $i \in \{1, \dots, n\}$  (in order) do
14:    $S_{\text{reliable}} \leftarrow \{s \in \{s_1, \dots, s_{i-1}\} : \|v(s)\|_2 > 2\delta\}$ 
15:    $S_{\text{-reliable}} \leftarrow \{s_1, \dots, s_{i-1}\} \setminus S_{\text{reliable}}$ 
16:    $d_{\min} \leftarrow \min_{s \in S_{\text{reliable}}} \text{vdist}(v(s_i), v(s))$  ▷ see Algo. 1,  $d_{\min} = \infty$  for  $S_{\text{reliable}} = \emptyset$ 
17:   if  $d_{\min} < \delta$  then
18:      $\omega(s_i) \leftarrow \omega(\text{argmin}_{s \in S_{\text{reliable}}} \text{vdist}(v(s_i), v(s)))$  ▷  $s_i$  added to aggregate which contains closest vector
19:   else
20:      $d_{\min} \leftarrow \min_{s \in S_{\text{-reliable}}} \text{vdist}(v(s_i), v(s))$  ▷ try again with less reliable vectors
21:     if  $d_{\min} < \delta$  then
22:        $\omega(s_i) \leftarrow \omega(\text{argmin}_{s \in S_{\text{-reliable}}} \text{vdist}(v(s_i), v(s)))$ 
23:     else
24:        $m \leftarrow m + 1$ 
25:        $\omega(s_i) \leftarrow m$  ▷  $s_i$  added to new aggregate
26:     end if
27:   end if
28: end for
29: return  $\omega$ 

```

Definition 25. We call a partition Ω ε -almost exactly lumpable if:

$$\forall s, s' \in S \text{ s.t. } \omega(s) = \omega(s') : \sum_{\rho \in \Omega} \left| \sum_{r \in \rho} P(r, s) - \sum_{r \in \rho} P(r, s') \right| \leq \varepsilon$$

We use the same definition for CTMCs with P replaced by Q .

The summed incoming probabilities to two states in the same aggregate from another aggregate are not required to be identical anymore as in Definition 16, but they are close to being identical.

Proposition 26. Given a partition Ω of the state space of a DTMC or CTMC which is ε -almost exactly lumpable, let $\alpha(s) = \frac{1}{|\omega(s)|}$ and $\Pi = A P A$ or $\Theta = A Q A$. Then

$$\|\Pi A - A P\|_\infty \leq |\Omega| \cdot \frac{\max_{\rho \in \Omega} |\rho|}{\min_{\rho \in \Omega} |\rho|} \cdot \varepsilon \quad \text{or} \quad \|\Theta A - A Q\|_\infty \leq |\Omega| \cdot \frac{\max_{\rho \in \Omega} |\rho|}{\min_{\rho \in \Omega} |\rho|} \cdot \varepsilon$$

Hence, given an algorithm which takes ε as input and outputs an ε -almost exactly lumpable partition, we can choose some ε which guarantees a desired error bound $\|\Pi A - A P\|_\infty$ in advance without actually running the algorithm and calculating $\|\Pi A - A P\|_\infty$ for the resulting partition. In practice, the bound in Proposition 26 always seemed to be much larger than the actual value of $\|\Pi A - A P\|_\infty$, though. Note that we have no equivalent result for the previously presented SVD partitioning algorithm. While Algorithm 2 also takes a (different) input parameter ε used for cutting off the smallest singular values (see (14)), it seems to be difficult to derive a bound on $\|\Pi A - A P\|_\infty$ depending on ε .

Proof of Proposition 26. We have

$$\begin{aligned} \| \Pi A - AP \|_\infty &\stackrel{\text{cf. (5)}}{=} \max_{\rho \in \Omega} \sum_{\sigma \in \Omega} \sum_{s \in \sigma} \left| \alpha(s) \Pi(\rho, \sigma) - \sum_{r \in \rho} \alpha(r) P(r, s) \right| \\ &\stackrel{\Pi = APA}{=} \max_{\rho \in \Omega} \sum_{\sigma \in \Omega} \sum_{s \in \sigma} \left| \frac{1}{|\sigma|} \sum_{r \in \rho} \frac{1}{|\rho|} \sum_{s' \in \sigma} P(r, s') - \sum_{r \in \rho} \frac{1}{|\rho|} P(r, s) \right| \\ &= \max_{\rho \in \Omega} \sum_{\sigma \in \Omega} \sum_{s \in \sigma} \left| \frac{1}{|\sigma|} \frac{1}{|\rho|} \sum_{s' \in \sigma} \sum_{r \in \rho} P(r, s') - \frac{1}{|\sigma|} \frac{1}{|\rho|} \sum_{s' \in \sigma} \sum_{r \in \rho} P(r, s) \right| \\ &= \max_{\rho \in \Omega} \sum_{\sigma \in \Omega} \frac{1}{|\sigma|} \frac{1}{|\rho|} \sum_{s \in \sigma} \left| \sum_{s' \in \sigma} \sum_{r \in \rho} (P(r, s') - P(r, s)) \right| \end{aligned}$$

By pulling the absolute value into the sum $\sum_{s' \in \sigma}$, we get

$$\begin{aligned} \| \Pi A - AP \|_\infty &\leq \max_{\rho \in \Omega} \sum_{\sigma \in \Omega} \frac{1}{|\sigma|} \frac{1}{|\rho|} \sum_{s \in \sigma} \sum_{s' \in \sigma} \left| \sum_{r \in \rho} (P(r, s') - P(r, s)) \right| \\ &\leq \sum_{\sigma \in \Omega} \frac{1}{|\sigma|} \sum_{s \in \sigma} \sum_{s' \in \sigma} \max_{\rho \in \Omega} \frac{1}{|\rho|} \left| \sum_{r \in \rho} P(r, s') - \sum_{r \in \rho} P(r, s) \right| \\ &\leq \sum_{\sigma \in \Omega} \frac{1}{|\sigma|} \sum_{s \in \sigma} \sum_{s' \in \sigma} \frac{1}{\min_{\rho \in \Omega} |\rho|} \underbrace{\max_{\rho \in \Omega} \left| \sum_{r \in \rho} P(r, s') - \sum_{r \in \rho} P(r, s) \right|}_{\leq \varepsilon} \\ &\leq \sum_{\sigma \in \Omega} \frac{|\sigma|}{\min_{\rho \in \Omega} |\rho|} \cdot \varepsilon \leq |\Omega| \cdot \frac{\max_{\rho \in \Omega} |\rho|}{\min_{\rho \in \Omega} |\rho|} \cdot \varepsilon \end{aligned} \tag{16}$$

The same calculation holds for the continuous time case. Also note that the bound can be improved to

$$\| \Pi A - AP \|_\infty \leq |\Omega| \cdot \frac{\max_{\rho \in \Omega} |\rho| - 1}{\min_{\rho \in \Omega} |\rho|} \cdot \varepsilon \tag{17}$$

by noting that the double sum $\sum_{s \in \sigma} \sum_{s' \in \sigma}$ in (16) actually sums over elements which are zero for $s = s'$. \square

Remark. The bound given in Proposition 26 cannot be significantly improved. We can give a series of examples of ε -almost exactly lumpable partitions Ω with $|\Omega| = m$, $|S| = n = 2m$, $\varepsilon = \frac{1}{2m} = \frac{1}{n}$ and $\| \Pi A - AP \|_\infty = |\Omega| \cdot \frac{\varepsilon}{2} = \frac{1}{4}$. Consider the state space $S = \{1, \dots, n\}$ with $n = 2m$, the partition $\Omega = \{\{1, 2\}, \{3, 4\}, \dots, \{2m-1, 2m\}\}$ and $\alpha(s) = \frac{1}{2} = \frac{1}{|\omega(s)|}$ for all s . Define $P(1, 2s) = \frac{2+\varepsilon n}{2n}$ and $P(1, 2s-1) = \frac{2-\varepsilon n}{2n} > 0$ for $s \in \{1, \dots, m\}$, and $P(r, s) = \frac{1}{n}$ for $r \geq 2$ and $s \in S$. This partition is ε -almost exactly lumpable. Indeed, consider two states s', s'' in the same aggregate. Then, w.l.o.g., we have $s' = 2s$ and $s'' = 2s-1$ for some $s \in \{1, \dots, m\}$. Hence

$$\begin{aligned} \sum_{\rho \in \Omega} \left| \sum_{r \in \rho} P(r, s') - \sum_{r \in \rho} P(r, s'') \right| &= \sum_{\rho \in \Omega} \left| \sum_{r \in \rho} P(r, 2s) - \sum_{r \in \rho} P(r, 2s-1) \right| \\ &= \left| \frac{2+\varepsilon n}{2n} + \frac{1}{n} - \frac{2-\varepsilon n}{2n} - \frac{1}{n} \right| + \underbrace{\sum_{\rho \in \Omega \setminus \{\{1, 2\}\}} \left| \sum_{r \in \rho} \frac{1}{n} - \sum_{r \in \rho} \frac{1}{n} \right|}_0 = \frac{2\varepsilon n}{2n} = \varepsilon \end{aligned}$$

Now, consider $r, s \in \{1, \dots, m\}$. Setting $\rho = \{2r-1, 2r\}$, $\sigma = \{2s-1, 2s\}$, we have

$$\Pi(\rho, \sigma) = \frac{1}{2} \cdot (P(2r-1, 2s-1) + P(2r-1, 2s)) + \frac{1}{2} \cdot (P(2r, 2s-1) + P(2r, 2s)) = \frac{2}{n}$$

Hence, it holds that

$$\begin{aligned} \| \Pi A - AP \|_\infty &= \max_{\rho \in \Omega} \sum_{\sigma \in \Omega} \sum_{s \in \sigma} \left| \alpha(s) \Pi(\rho, \sigma) - \sum_{r \in \rho} \alpha(r) P(r, s) \right| \\ &= \max_{\rho \in \Omega} \sum_{s=1}^m \left(\left| \alpha(2s-1) \cdot \frac{2}{n} - \sum_{r \in \rho} \alpha(r) P(r, 2s-1) \right| + \left| \alpha(2s) \cdot \frac{2}{n} - \sum_{r \in \rho} \alpha(r) P(r, 2s) \right| \right) \\ &= \sum_{s=1}^m \left(\left| \frac{1}{2} \cdot \frac{2}{n} - \sum_{r \in \{1, 2\}} \frac{1}{2} P(r, 2s-1) \right| + \left| \frac{1}{2} \cdot \frac{2}{n} - \sum_{r \in \{1, 2\}} \frac{1}{2} P(r, 2s) \right| \right) \\ &= \sum_{s=1}^m \left(\left| \frac{1}{2} \cdot \frac{2}{n} - \frac{4-\varepsilon n}{4n} \right| + \left| \frac{1}{2} \cdot \frac{2}{n} - \frac{4+\varepsilon n}{4n} \right| \right) = \sum_{s=1}^m \frac{\varepsilon}{2} = m \cdot \frac{\varepsilon}{2} = |\Omega| \cdot \frac{\varepsilon}{2} \end{aligned}$$

Therefore, we cannot drop the dependence on $|\Omega|$ in the bound given in [Proposition 26](#). Actually, the improved bound given in [\(17\)](#) is tight in this case as we have $\min_{\rho \in \Omega} |\rho| = \max_{\rho \in \Omega} |\rho| = 2$.

We now develop an algorithm which finds an ε -almost exactly lumpable partition as a counterpart to the SVD approach for almost aggregatable partitions. The algorithm works for both DTMCs and CTMCs (we give the DTMC version, but for CTMCs, P only has to be replaced by Q). For a given ε , the algorithm should find a partition which is as coarse as possible and still satisfies ε -almost exact lumpability. Note that in general, there is no unique coarsest ε -almost exactly lumpable partition. However, there always is a unique coarsest *exactly lumpable* partition which may be found by successive refinement of the partition $\Omega = \{S\}$ (this was shown in [\[8\]](#), where the partition refinement approach is mentioned on p. 269). We can thus hope to get good results by using a successive refinement algorithm for ε -almost exact lumpability as well, and we use this approach in [Algorithm 3](#). It does not necessarily find a partition with as few aggregates as possible, but has performed well in experiments.

Algorithm 3 Calculating almost exactly lumpable partitions

Input: a Markov chain, defined via its transition matrix P on state space S ,
and the parameter ε (a generator matrix Q can be used instead of P)

Output: an aggregation function ω
whose corresponding partition is ε -almost exactly lumpable

```

1:  $\omega^{(1)} \leftarrow ((s \in S) \mapsto 1)$  ▷ aggregation function
2:  $i \leftarrow 1$  ▷ iteration counter
3:  $m \leftarrow 1$  ▷ number of aggregates
4: repeat
5:    $m_{\text{old}} \leftarrow m$  ▷ saves number of old aggregates
6:    $m \leftarrow 0$  ▷ counts number of new aggregates
7:   for all  $j \in \{1, \dots, m_{\text{old}}\}$  do ▷ loop over old/target aggregates
8:     for all  $s \in \{r \in S : \omega^{(i)}(r) = j\}$  do ▷ loop over states in same aggregate
9:        $\text{inc}(s) \leftarrow \vec{0} \in \mathbb{R}^{m_{\text{old}}}$ 
10:      for all  $k \in \{1, \dots, m_{\text{old}}\}$  do ▷ loop over potential splitter aggregates
11:         $\text{inc}(s)_k \leftarrow \sum_{r \in S : \omega^{(i)}(r) = k} P(r, s)$  ▷ incoming probability from agg.  $k$  to state  $s$ 
12:      end for
13:    end for
14:     $C \leftarrow \text{cluster}(\{r \in S : \omega^{(i)}(r) = j\}, \text{inc}, \varepsilon)$  ▷ see below
15:    for all  $\sigma \in C$  do ▷ loop over clusters
16:      for all  $s \in \sigma$  do
17:         $\omega^{(i+1)}(s) \leftarrow m + 1$  ▷ states in  $\sigma$  are assigned to the same aggregate
18:      end for
19:       $m \leftarrow m + 1$  ▷ increment aggregate number
20:    end for
21:  end for
22:   $i \leftarrow i + 1$ 
23: until  $m_{\text{old}} = m$  ▷ stop when no aggregates were split
24: return  $\omega^{(i)}$ 

```

The idea of the algorithm is as follows: we start with the initial partition $\Omega = \{S\}$ (represented in [Algorithm 3](#) by the aggregation function $\omega : S \rightarrow \mathbb{N}$ which maps every state to aggregate 1). Ω is then successively refined. At every refinement step, for every aggregate $\sigma \in \Omega$ and for all states $s \in \sigma$, we construct vectors of incoming probabilities

$$\text{inc}(s) = \left(\sum_{r \in \Omega_1} P(r, s), \dots, \sum_{r \in \Omega_m} P(r, s) \right)^T \in \mathbb{R}^m$$

where m is the current number of aggregates in Ω . For an ε -almost exactly lumpable partition, it needs to hold that the entries of the vectors $\text{inc}(s)$ and $\text{inc}(s')$ are close together for s, s' in the same aggregate σ . Actually, by [Definition 25](#), we have that the current partition Ω is ε -almost exactly lumpable if, and only if, $\|\text{inc}(s) - \text{inc}(s')\|_1 \leq \varepsilon$. If this is not the case, the algorithm therefore proceeds with the refinement by partitioning the states in $\sigma \in \Omega$ into smaller aggregates such that $\|\text{inc}(s) - \text{inc}(s')\|_1 \leq \varepsilon$ for two states s, s' in the same aggregate in the resulting refined partition.

The procedure stops when an ε -almost exactly lumpable partition is found, at the latest when every aggregate consists of a single state. The refinement step amounts to clustering points in \mathbb{R}^m such that the maximal $\|\cdot\|_1$ -distance between any pair of points in a cluster is at most ε . The method $\text{cluster}(T, f, \varepsilon)$ takes a subset of states $T \subseteq S$, a function $f : T \rightarrow \mathbb{R}^k$ and a parameter $\varepsilon > 0$ as input. The output is a partition C of T such that for any cluster $\sigma \in C$ and any two states $s, s' \in \sigma$, we have that $\|f(s) - f(s')\|_1 \leq \varepsilon$. Of course, the method should try to return as few clusters as possible, but our Python implementation does not guarantee an optimal solution. We usually apply the hierarchical clustering utilities offered by the SciPy Python package, namely the method

`scipy.cluster.hierarchy.fclusterdata`. We will see that this method performs well in our experiments, but we also sometimes switch to a greedy strategy to speed up computations.

The greedy clustering algorithm proceeds as follows: we iterate over all the vectors which we want to cluster. Every cluster is assigned a so-called anchor vector, and when processing a new vector, we check if it has $\|\cdot\|_1$ -distance of at most $\frac{\epsilon}{2}$ to any of the anchor vectors of the clusters formed by previously processed vectors. If this is the case, we assign the vector to the cluster of the first such anchor vector which we find. Otherwise, a new cluster with the current vector as anchor vector is created.

To conclude this section, we will briefly discuss the runtime of Algorithm 3. Denote by m the number of aggregates returned by the algorithm (which is not known in advance). The outer loop (lines 4 to 23) runs through at most m iterations. The loops in lines 7 to 8 lead to n executions of the inner loop on lines 10 to 12. Lines 10 to 12, in turn, run in time $\mathcal{O}(n)$ since the loop in line 10 iterates over all aggregates, and line 11 then calculates a sum over all states in the respective aggregate. Therefore, the loops in lines 7 to 13 contribute a runtime of $\mathcal{O}(n^2)$ per iteration of the outer loop.

The runtime of line 14 depends on the clustering algorithm which is used. In our implementation, `scipy.cluster.hierarchy.fclusterdata` runs in time $\mathcal{O}(mn^2)$ because it gets at most n vectors, one per state, as input² and because the vectors are of dimension at most m . Since line 14 is executed within the loop on line 7 (which does at most m iterations), this contributes a runtime of $\mathcal{O}(m^2n^2)$ per iteration of the outer loop (larger than the $\mathcal{O}(n^2)$ of lines 7 to 13). Lines 15 to 20 run in $\mathcal{O}(n)$, so these are faster than line 14 and do not add to the runtime. As the outer loop runs at most m times, we arrive at a total runtime of $\mathcal{O}(m^3n^2)$.

Comparing with the runtimes for exact calculation of transient distributions p_k (respectively p_i), we see that, asymptotically for large k and n , applying Algorithm 3 makes sense if $m^3 \ll k$ (respectively $m^3 \ll qt$ where q is the maximal exit rate of all states in the CTMC). However, the runtime bound of $\mathcal{O}(m^3n^2)$ seems to be rarely achieved in practice, so Algorithm 3 can still be useful if we do not have $m^3 \ll k$.

If we implement the greedy clustering instead of using `scipy.cluster.hierarchy.fclusterdata`, then the clustering takes time $\mathcal{O}(m^2n)$ (because only distances to cluster anchor vectors are computed, instead of distances to all vectors), reducing the overall runtime to $\mathcal{O}(m^4n)$ (again, in practice, the algorithm seemed to be significantly faster than that — for example, we usually observed that the outer loop of Algorithm 3 was executed a number of times which was significantly lower than m , e.g. by a factor of 100).

6. Experiments

In this section, we will compare the performance of SVDsgn, SVDseba, SVDdir (Algorithm 2), and Algorithm 3 on a selection of Markov chains. By performance comparison, we mean comparing the error bound given by $\|IIA - AP\|_\infty$ (or $\|\Theta A - AQ\|_\infty$) resulting from the aggregations returned by the different algorithms — the lower, the better. We will not compare initial errors, as neither the SVD algorithms nor Algorithm 3 take into account the initial distribution. The development of efficient algorithms taking into account both initial and dynamical error is left for future work.

We first look at the setting for which the SVD algorithm variants were designed: almost aggregatable Markov chains. These are easy to generate randomly and we can compare the performance of the different algorithms. Afterwards, we will see some of the examples used in [5] as well as an example derived from a stochastic process algebra model which allows for an exactly lumpable partition, so we will also see a setting for which Algorithm 3 was designed. By default, we will calculate the α distributions as in (15), and Π (or Θ) will be set to $\Pi = APA$ (or $\Theta = AQA$).

For some of the models, we used faster versions of SVDdir (Algorithm 2) and Algorithm 3 as already briefly mentioned before: for the fast variant of SVDdir, instead of using (14) to determine the number of singular values considered, we simply pass different fixed numbers of singular values to be calculated to the algorithm which allows us to bound the runtime (as we only need to compute a partial decomposition), and which will result in different aggregations depending on the number passed. The fast variant of Algorithm 3 switches to the greedy clustering method described after the introduction of Algorithm 3 when $d^2 m > 4 \cdot 10^6$, where d is the number of vectors (of incoming probabilities) to be clustered and m is the current number of aggregates (other choices than $4 \cdot 10^6$ are of course possible, depending on how one wants to control the runtime). In line 14 of Algorithm 3, the algorithm will thus dynamically decide which clustering method to use depending on whether `scipy.cluster.hierarchy.fclusterdata` is expected to have a high runtime. The figures using the fast algorithm variants will have captions specifying that SVDdir (fast) or Algorithm 3 (fast) were used.

6.1. Almost aggregatable Markov chains

In Fig. 2, we consider a range of randomly generated almost aggregatable DTMCs. $\|IIA - AP\|_\infty$ of the aggregation returned by the algorithms (run with different input parameters) is plotted against the number of aggregates which are found (which depends on the input parameter ϵ ; the smaller ϵ , the more aggregates). We can see that the SVD variants (except for SVDseba) perform better than Algorithm 3 for these almost aggregatable Markov chains, which is no surprise. In addition, the improved stability of SVDdir clearly pays off in comparison to SVDsgn: We can see a sharp drop in the error bounds around 20 aggregates, which was the number

² For the runtime, see the SciPy documentation at <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html>.

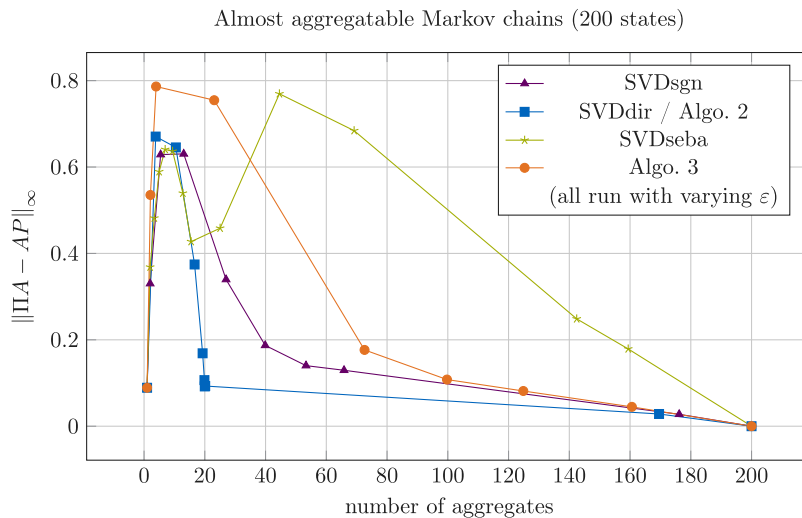


Fig. 2. SVDsgn, SVDdir, SVDseba and Algorithm 3 executed on 100 randomly generated almost aggregatable DTMCs with 200 states, 20 aggregates and a probability of 0.5 to have no transition between a particular pair of aggregates. The almost aggregatable DTMCs were obtained by random perturbation (with a magnitude of 0.002) of the transition matrix of a randomly generated aggregatable DTMC. Each plotted point is an average resulting from running the algorithms with a particular fixed input parameter ε on the 100 DTMCs.

of aggregates in the almost aggregatable partition. For SVDsgn, the drop is more a gradual decrease in the error bounds. Algorithm 3 does not identify the almost aggregatable partition and only reaches a similar error bound level for around 120 aggregates.

Note that the error bound of around 0.09 for a single aggregate (the leftmost point in the plots in Fig. 2) means that the distribution $\alpha_{\{S\}}$ obtained via (15) is close to being stationary, as we have $\Pi = (1) \in \mathbb{R}^{1 \times 1}$ in this case and hence

$$0.09 \approx \|\Pi A - AP\|_{\infty} = \left\| \alpha_{\{S\}}^T - \alpha_{\{S\}}^T P \right\|_{\infty}$$

The closeness of $\alpha_{\{S\}}$ to stationarity is no big surprise for randomly generated Markov chains as these tend to have stationary distributions which are closer to the uniform distribution when compared to Markov chains with more structure.

SVDseba performs similarly to SVDdir for a low number of aggregates, but there is a sudden change around 20 aggregates when SVDseba starts to perform worse than all other algorithms. This is due to the fact that we limited the maximum number of iterations of the SEBA algorithm (see [17, Algorithm 3.1], we took the MATLAB code given in [17] and translated it into Python) to 300 iterations because of its high runtime. Regardless of the number of maximum iterations, we could never observe SVDseba performing significantly better than SVDdir in all our experiments. The latter is therefore a good alternative. The better performance of SVDdir might be due to its specificity for the given problem. The SEBA algorithm only tries to find a sparse basis for the row space of the first m rows of the matrix V in the singular value decomposition. It was designed for general applications, and does not exploit the fact that the vectors $v(r)$ and $v(s)$ are approximate multiples of one another for almost aggregatable DTMCs when r and s belong to the same aggregate.

In the following experiments, we will focus on SVDdir as the best compromise between numerical stability and speed among the SVD approaches. We looked at some of the models which were already used in the experimental section of [5] where the error bounds which were further developed in this paper were first introduced. In [5], a very simple aggregation strategy was used: the aggregation algorithm started with singleton clusters (every state is its own cluster), then iterated over all transitions in the model, in descending order of the transition probability, and merged the clusters of the two connected states, unless this would result in clusters above a user-defined maximal cluster size. This procedure will in general result in a very high error bound $\|\Pi A - AP\|_{\infty}$, but [5] then used a dynamic aggregate-splitting approach while calculating transient distributions. The precalculated clusters were stored during the whole computation, but whenever a certain cluster had a transient probability above a given threshold at some point during the time evolution, it was split into singletons. The singletons were merged again into the old precalculated cluster if the overall probability of being in that cluster would fall below the threshold at a later point. As can be seen by looking at Theorem 4(i), this will also result in a low overall error bound, as the scalar product will only become large if both the transient probability of an aggregate and the error caused by that aggregate are large. In this paper, we wanted to analyze whether a significant state space reduction is already possible without dynamic aggregate-splitting if we use different aggregation algorithms.

6.2. A prokaryotic gene expression model

In Fig. 3, we considered a prokaryotic gene expression model [19] with a state space of 44k states, which is the limit of what the fast SVDdir algorithm could handle on the machine which we used in all our experiments (single-threaded execution on an Intel Core i7-1260P CPU with a maximum frequency of 4.7 GHz). The graph of the SVDdir algorithm stops at around 8k aggregates due

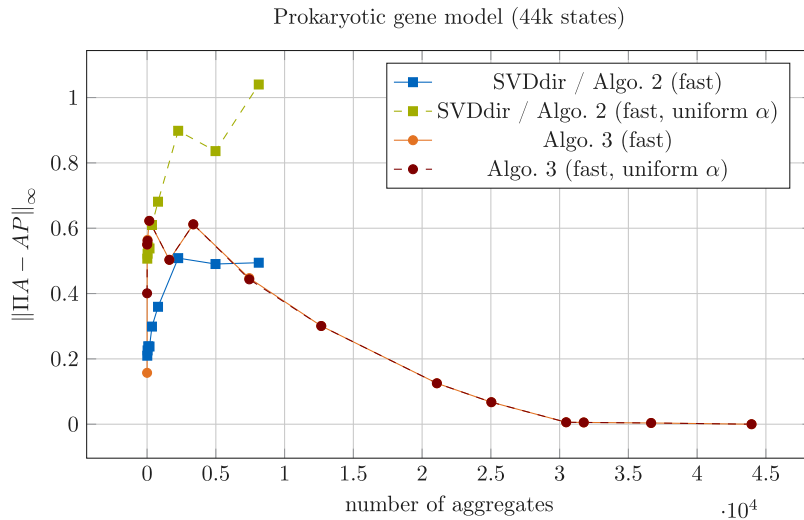


Fig. 3. SVDdir and Algorithm 3 executed on a prokaryotic gene expression model already used in the experiments in [5], originally from [19]. The maximum population size was set to 5, resulting in 43 957 states. The CTMC was uniformized using the maximal exit rate 16.78 as uniformization rate.

to its runtime of $\mathcal{O}(n^2l)$ (with n the number of states and l the number of singular values considered — the more values considered, the more aggregates the algorithm will usually return) which did not allow us to compute more points. We can see that Algorithm 3 can find state space reductions down to around 30k states with very low error bounds (the exact values here are 30 476 aggregates with $\|\Pi A - AP\|_\infty \approx 0.0060$) which corresponds to a state space reduction to around 70%. Below that, the error bound starts to increase gradually, resulting in aggregations where the error bounds get too high to be useful in practice. SVDdir performed only slightly better for a low number of aggregates, but the resulting error bounds are still too high for practical purposes.

Fig. 3 also compares two ways of setting the α distributions: the default method of (15) and setting $\alpha(s) = \frac{1}{|\omega(s)|}$, resulting in uniform α distributions and referred to as uniform α in the remaining paper. The method makes no difference to the error bound achieved by Algorithm 3 which can be explained by the fact that Algorithm 3 anyway identifies aggregates where the sums of incoming probabilities used in (15) are close to being identical (note that the orange and red dashed curve are almost exactly on top of each other in Fig. 3). However, the error bound achieved by SVDdir (Algorithm 2) is much better if (15) is used, because SVDdir, in contrast to Algorithm 3, can also detect aggregations where the α distributions are not uniform.

6.3. The Lotka–Volterra model

In Fig. 4, we consider the well-known Lotka–Volterra model (see, e.g., [20]) which can be used to represent the number of predators and prey in a simple setting. We can see that the error bounds achieved by Algorithm 3 are comparatively low while using SVDdir does not result in very helpful aggregations in this case. Also note that $\|\Pi A - AP\|_\infty \approx 0.0092$ at a single aggregate (i.e. all states in one aggregate), which is quite low. As discussed before, this implies that $\alpha_{\{S\}}$ is very close to being stationary, and $\alpha_{\{S\}}$ is the uniform distribution if we use uniform α . This reduction is only useful in practice if the initial distribution is close to the uniform distribution. A better error bound is only achieved at an aggregation with 9 972 aggregates and $\|\Pi A - AP\|_\infty \approx 0.0057$ which is only a state space reduction to 98% of the original size.

Note that the low error bound at a single aggregate is also due to the fact that Fig. 4 considered a uniformized CTMC with a relatively high uniformization rate of 2078. $\|\Pi A - AP\|_\infty$, where P and Π are the uniformized versions of the generator Q of the original CTMC and the aggregated evolution matrix Θ , corresponds to $\frac{1}{q}$ times $\|\Theta A - AQ\|_\infty$ where q is the uniformization rate. Hence, on the original time scale of the CTMC, the error bound would actually be 2078 times higher, which make the mentioned bounds from above not very useful on the original time scale in practice. In addition, the high uniformization rate also causes the uniform distribution to be close to stationarity for the uniformized DTMC, which also explains the low error bound at a single aggregate.

6.4. A stochastic process algebra model

Next to the models from [5,19,20], we also considered a compositional stochastic process algebra model, the RSVP model from [21], at which we take a closer look. It comprises a lower network channel submodel with capacity for M calls, an upper network channel submodel with capacity for N calls, and a number of identical mobile nodes which request resources for calls at a constant rate. Due to the mobile node symmetry in the model specification, a lossless reduction is possible for this model. Comparing the different algorithms in Fig. 5, we see that only Algorithm 3 identifies the partition which results in a lossless reduction and which

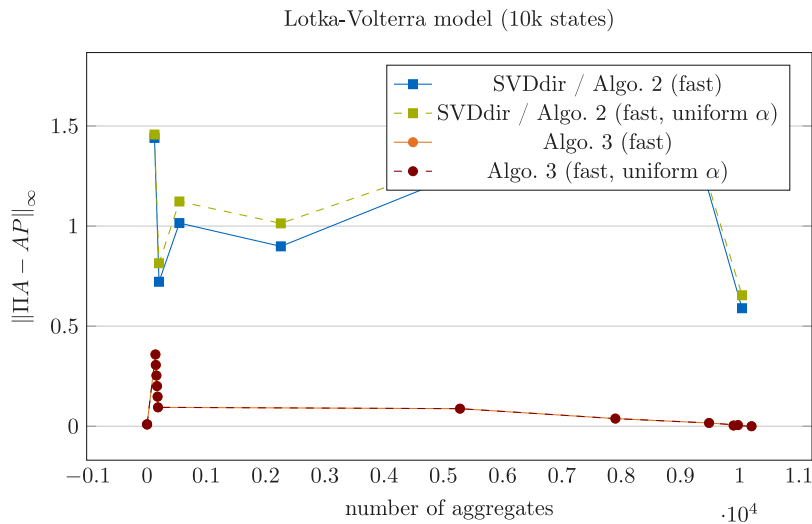


Fig. 4. SVDdir and Algorithm 3 executed on the Lotka–Volterra model already used in the experiments in [5], described in more detail e.g. in [20]. The maximal species population size was set to 100, resulting in 10 201 states. The CTMC was uniformized using the maximal exit rate 2078 as uniformization rate.

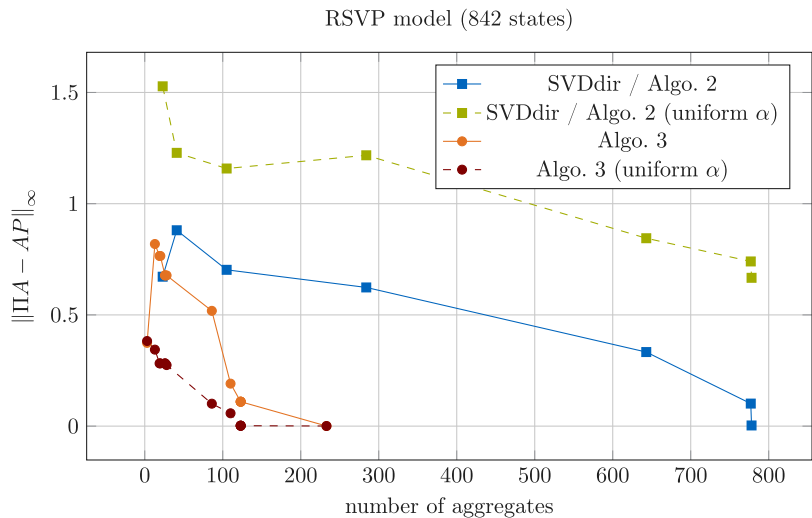


Fig. 5. SVDdir and Algorithm 3 executed on the DTMC arising when uniformizing the RSVP model with $M = 7$, $N = 5$ and 3 mobile nodes, resulting in a total of 842 states. The uniformization rate was set to the maximal exit rate among all states, which is 30.01 in this case.

is exactly lumpable: the error bound $\|\Pi A - AP\|_\infty$ is equal to 0 for 234 aggregates. SVDdir performs much worse. Algorithm 3 also finds a reduction to 123 aggregates (reduction to 15%) at an error of $\|\Pi A - AP\|_\infty \approx 0.0019$.

In Fig. 6, we apply Algorithm 3 directly to the CTMC corresponding to the RSVP model and compare setting $\Theta = AQA$ with the median-based scheme from Section 4.3. The error bounds are similar in magnitude for the usual way of setting $\Theta = AQA$ and for the median-based scheme. But we also see that the error bounds are much higher than for the uniformized version (compare with Fig. 5). This is because $\|\Pi A - AP\|_\infty$, where P and Π are the uniformized versions of Q and Θ , corresponds to $\frac{1}{q}$ times $\|\Theta A - AQ\|_\infty$ where q is the uniformization rate, which is set to the maximal exit rate 30.01 in this case. In addition, Fig. 6 demonstrates that uniform α , i.e. setting $\alpha(s) = \frac{1}{|\varphi(s)|}$, works better than proportional α as in (15) for Algorithm 3 for this model, while we saw few differences for the other models.

6.5. Runtime evaluation

We performed a runtime evaluation on randomly generated Markov chains which are almost exactly lumpable in order to compare our theoretical runtime analysis with measurements. In Table 2, we compare the speed of using Algorithm 3 and computing \tilde{p}_i to the speed of computing p_i directly (using the Fox-Glynn method [15,22]). Our theoretical runtime analysis was done for dense

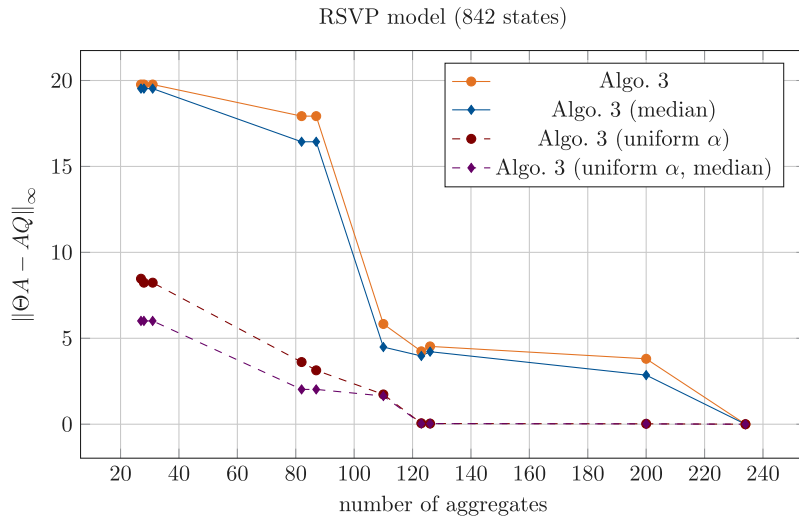


Fig. 6. Algorithm 3 executed on the CTMC arising from the RSVP model with $M = 7$, $N = 5$ and 3 mobile nodes, resulting in a total of 842 states.

Table 2

Speed-up (the lower the value, the higher the speed-up) resulting from using Algorithm 3. The depicted values correspond to the time needed to aggregate a CTMC using Algorithm 3 and then computing the approximate transient distribution \tilde{p}_i , at different points in time, divided by the time needed to compute the transient distribution exactly. For every data point, Algorithm 3 was executed on 200 randomly generated almost exactly lumpable CTMCs with 1000 states and a varying number of aggregates. The almost exactly lumpable CTMCs were obtained by random perturbation (with a magnitude of $5 \cdot 10^{-6}$) of the generator matrix of a randomly generated exactly lumpable CTMC. Each plotted value is an average over the 200 randomly sampled CTMCs. We rejected 0.9% of the randomly generated CTMCs where Algorithm 3 did not correctly identify the generated almost exactly lumpable partition.

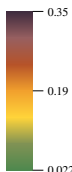
Aggregates \ Time	Time				
	100	200	400	600	800
5	1.22	1.05	0.77	0.63	0.53
7	1.28	1.06	0.79	0.65	0.5
10	1.43	1.2	0.87	0.74	0.61
15	1.49	1.22	0.91	0.73	0.61
20	1.54	1.31	0.98	0.79	0.65



Table 3

The average error bound computed using Theorem 5(iii) for the 200 randomly generated CTMCs from Table 2.

Aggregates \ Time	Time				
	100	200	400	600	800
5	0.022	0.044	0.088	0.13	0.18
7	0.026	0.052	0.1	0.15	0.21
10	0.031	0.061	0.12	0.18	0.25
15	0.038	0.075	0.15	0.23	0.3
20	0.044	0.087	0.17	0.26	0.35



transition and generator matrices, so we randomly generated dense generator matrices of almost exactly lumpable CTMCs. As can be seen in Table 2, Algorithm 3 speeds up the computation if the reduction in the size of the state space is big enough and if we compute transient distributions at a point in time which is large enough, which is consistent with the theory.

Remark. While the computation of the exact transient distributions relies almost solely on highly optimized linear-algebra routines which use compiled code, we only implemented Algorithm 3 in Python without any particular optimizations. It is therefore probable that we could further improve the runtime by using a more performance-oriented and compiled language. Table 2 shows that even with Python, we can already achieve a speed-up in some cases.

Table 4

The average actual error $\|p_t - \tilde{p}_t\|$, for the 200 randomly generated CTMCs from Table 2. Note that all values have a similar magnitude. The colors are only meant to show that no clear pattern exists, and not that some of the values are much worse than the others. The differences are rather due to random fluctuations.

Aggregates \ Time	100	200	400	600	800
5	$7.4 \cdot 10^{-5}$	$8 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	$7.1 \cdot 10^{-5}$	$7.2 \cdot 10^{-5}$
7	$7.1 \cdot 10^{-5}$	$7.3 \cdot 10^{-5}$	$7.6 \cdot 10^{-5}$	$7.2 \cdot 10^{-5}$	$6.3 \cdot 10^{-5}$
10	$8.8 \cdot 10^{-5}$	$8.2 \cdot 10^{-5}$	$7.4 \cdot 10^{-5}$	$8.9 \cdot 10^{-5}$	$7.7 \cdot 10^{-5}$
15	$7.7 \cdot 10^{-5}$	$7.1 \cdot 10^{-5}$	$7.3 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	$7.1 \cdot 10^{-5}$
20	$7.2 \cdot 10^{-5}$	$7.7 \cdot 10^{-5}$	$8.2 \cdot 10^{-5}$	$8.2 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$

Tables 3 and 4 show the average error bounds and actual errors which resulted from the aggregation. We chose the magnitude of perturbation compared to an exactly lumpable Markov chain in such a way that the computed error bounds are on the boundary between useful and useless in practice (even though this can depend a lot on the intended application). Here, we used the linear bound given by Theorem 5(iii) which could be further improved by Theorem 5(i). If we look at Table 4, we see that the actual errors are significantly lower in these cases. This is due to the fact that the tightness result essentially only tells us that the rate at which the error grows will be identical to the error bound near time 0.

The initial distributions were chosen to be compatible with the aggregations in these examples, as Algorithm 3 does not take initial distributions into account. However, there is an easy adaption to Algorithm 3 which guarantees that no initial error is made when the actual initial distribution is concentrated on a single state: we can simply start the algorithm with a predefined partition, where the initial state is its own aggregate, and all other states belong to a second aggregate. The algorithm will then only further subdivide these predefined aggregates.

We did the same runtime evaluation for Algorithm 2. According to the theory, we can speed up computations by using Algorithm 2 when the time point k for which transient distributions should be computed is larger than the number l of singular values which are computed. Calculating a partial singular value decomposition of a dense matrix where l singular values are computed takes $\mathcal{O}(ln^2)$ time, just as long as the clustering part performed by Algorithm 2. However, since the computation of the partial singular value decomposition is outsourced by the SciPy library to the compiled and highly optimized routine ARPACK [18], we find that the state clustering part, which is implemented in pure Python, takes longer by a factor of about 100 to 200. We could therefore only observe performance improvements compared to the exact calculation of transient distributions at impractically high values of k where the transient distributions are very close to the stationary distribution already. A different clustering method or code optimizations (and probably switching to a compiled language) would be necessary to achieve the speed-up predicted by the theory. Improving the runtime of the clustering part by a factor of 30 would result in a relative speed-up similar to the one depicted in Table 2.

6.6. Summary of results

Apart from the randomly generated almost aggregatable chains, Algorithm 3 performed better than SVDdir in all our experiments. Furthermore, we saw that Algorithm 3 can sometimes reduce the state space with an associated error bound low enough to be useful in practice, at size reductions in a wide range from down to e.g. 15% to 70% of the original size, the achieved size reduction depending very much on the model. As it only takes $\mathcal{O}(km^2)$ time to calculate transient distributions of the aggregated model with k the time horizon and m the number of aggregates, compared to $\mathcal{O}(kn^2)$ for the full model, such size reductions can significantly speed up the computation of transient distributions, if we ignore the time needed for aggregation (e.g. by a factor of around 44 for a reduction to 15% and a factor of around 2 for a reduction to 70%). If we also consider the time it takes to aggregate the model, the speed-up is less pronounced, but still significant in some settings, as demonstrated in Table 2. Hence, Algorithm 3 with its runtime of $\mathcal{O}(m^4n)$ for the fast variant (which is only a crude upper bound) can be a very useful tool in practice to check if a state space reduction is possible and to find a good partition resulting in a low error bound.

7. Conclusion and outlook

In this paper, we extended the error bounds originally derived in [5] to a more general and abstract setting where an aggregate no longer needs to be a group of states, and where each state within an aggregate can be assigned an individual weight, which need not be positive. Due to this generalization, the bounds can now be used in conjunction with almost any aggregation approach that has been considered in the literature so far. We also analyzed the meaning of these bounds for CTMCs, and thereby lay the groundwork for developing error bounds in more complex domains, such as systems with continuous state spaces. The paper showed that the presented error bounds are the best possible bounds in general for the difference between the transient distribution of an aggregated Markov chain and the original chain. Our analysis also pointed out a relation of the error bounds to existing lumpability concepts. Surprisingly, the general case for which the correct transient distributions of a Markov chain can be derived from the aggregated model (a dynamic-exact or exact aggregation) had only been considered in a small part of the existing literature (see

e.g. [10] or [14]), even though the concept of dynamic-exactness and exactness had already appeared under various names in the previous decades. In addition, we showed that the error bounds can also be applied to bound the distance of an approximated stationary distribution to stationarity, but not to the actual stationary distribution.

Calculating an aggregation which results in a good approximation of the original dynamics is difficult for general Markov chains, and we could not find a fast algorithm which identifies dynamic-exact aggregations in all cases, up to now. Instead, we compared two algorithms which identify two different, computationally tractable settings in which aggregations close to dynamic-exactness are possible and where the error bounds are low. The SVD algorithm from [7], augmented by us with the clustering by vector direction (i.e. SVDdir), had a much lower runtime than the SVD approach combined with SEBA from [17], while preserving the quality of the returned aggregations at the same time. For almost aggregatable Markov chains, the SVD algorithm is a good choice for identifying aggregates. However, in most of our experiments, we saw that Algorithm 3 (based on the concepts from [8]) performed better than the SVD variants, which makes it a promising alternative.

A detailed comparison to further possible aggregation algorithms would be necessary to fully evaluate accuracy and runtime of the different possibilities. However, from a theoretical perspective, this paper already established that the error bounds from [5] are a good tool to bound the aggregation error in much more general settings than originally envisioned, and first experiments showed that identifying almost exactly lumpable partitions with Algorithm 3 might be a good way to find suitable aggregations. In applications, we recommend a combination with the adaptive reaggregation and truncation techniques from [5], but replacing their base aggregation algorithm with one of our techniques, or even another scheme from the literature. As we focused on the theoretical side and the pure state grouping part, a study of this approach was beyond the scope of the present paper.

Another interesting topic for future work would be to develop an efficient algorithm which directly finds an approximate solution to $\Pi A = AP$ while taking the initial error into account at the same time. It is not clear, however, whether such an efficient algorithm exists at all.

CRedit authorship contribution statement

Fabian Michel: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Markus Siegle:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

- [1] H.A. Simon, A. Ando, Aggregation of variables in dynamic systems, *Econometrica* 29 (2) (1961) 111–138, <http://dx.doi.org/10.2307/1909285>.
- [2] J.G. Kemeny, J.L. Snell, *Finite Markov Chains*, Springer, 1976, URL <https://link.springer.com/book/9780387901923>.
- [3] G. Rubino, B. Sericola, A finite characterization of weak lumpable Markov processes, Part II: The continuous time case, *Stochastic Process. Appl.* 45 (1) (1993) 115–125, [http://dx.doi.org/10.1016/0304-4149\(93\)90063-A](http://dx.doi.org/10.1016/0304-4149(93)90063-A).
- [4] P. Buchholz, Exact and ordinary lumpability in finite Markov chains, *J. Appl. Probab.* 31 (1) (1994) 59–75, <http://dx.doi.org/10.2307/3215235>.
- [5] A. Abate, R. Andriushchenko, M. Češka, M. Kwiatkowska, Adaptive Formal Approximations of Markov Chains, *Perform. Eval.* 148 (102207) (2021) <http://dx.doi.org/10.1016/j.peva.2021.102207>.
- [6] P. Deufhard, M. Weber, Robust Perron cluster analysis in conformation dynamics, *Linear Algebra Appl.* 398 (2005) 161–184, <http://dx.doi.org/10.1016/j.laa.2004.10.026>.
- [7] A. Bittracher, C. Schütte, A probabilistic algorithm for aggregating vastly undersampled large Markov chains, *Physica D* 416 (132799) (2021) <http://dx.doi.org/10.1016/j.physd.2020.132799>.
- [8] P. Buchholz, Exact performance equivalence: An equivalence relation for stochastic automata, *Theoret. Comput. Sci.* 215 (1999) 263–287, [http://dx.doi.org/10.1016/S0304-3975\(98\)00169-8](http://dx.doi.org/10.1016/S0304-3975(98)00169-8).
- [9] F. Michel, M. Siegle, Markov chain aggregation with error bounds on transient distributions, in: A. Devos, A. Horváth, S. Rossi (Eds.), *Analytical and Stochastic Modelling Techniques and Applications*, Springer Nature, Switzerland, 2025, pp. 1–17, http://dx.doi.org/10.1007/978-3-031-70753-7_1.
- [10] A. Ganguly, T. Petrov, H. Koepl, Markov chain aggregation and its applications to combinatorial reaction networks, *J. Math. Biol.* 69 (3) (2014) 767–797, <http://dx.doi.org/10.1007/s00285-013-0738-7>.
- [11] G.H. Golub, C.F.V. Loan, *Matrix Computations*, fourth ed., Johns Hopkins University Press, 2013, <http://dx.doi.org/10.56021/9781421407944>.
- [12] H.L. Royden, *Real Analysis*, third ed., Collier Macmillan, 1988.
- [13] J. Ledoux, L. Truffet, Markovian bounds on functions of finite Markov chains, *Adv. in Appl. Probab.* 33 (2) (2001) 505–519, <http://dx.doi.org/10.1017/S0001867800010910>.
- [14] P. Buchholz, Bisimulation relations for weighted automata, *Theoret. Comput. Sci.* 393 (2008) 109–123, <http://dx.doi.org/10.1016/j.tcs.2007.11.018>.
- [15] B.L. Fox, P.W. Glynn, Computing Poisson probabilities, *Commun. ACM* 31 (4) (1988) 440–445, <http://dx.doi.org/10.1145/42404.42409>.
- [16] L.N. Trefethen, D. Bau, *Numerical Linear Algebra*, Twenty-fifth Anniversary ed., Society for Industrial and Applied Mathematics, 2022, <http://dx.doi.org/10.1137/1.9781611977165>.
- [17] G. Froyland, C.P. Rock, K. Sakellariou, Sparse eigenbasis approximation: Multiple feature extraction across spatiotemporal scales with application to coherent set identification, *Commun. Nonlinear Sci. Numer. Simul.* 77 (2019) 81–107, <http://dx.doi.org/10.1016/j.cnsns.2019.04.012>.

- [18] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK Users' Guide, Society for Industrial and Applied Mathematics, 1998, <http://dx.doi.org/10.1137/1.9780898719628>.
- [19] A.M. Kierzek, J. Zaim, P. Zielenkiewicz, The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression, *J. Biol. Chem.* 276 (11) (2001) 8165–8171, <http://dx.doi.org/10.1074/jbc.M006264200>.
- [20] D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions, *J. Phys. Chem.* 81 (25) (1977) 2340–2361, <http://dx.doi.org/10.1021/j100540a008>.
- [21] H. Wang, D.I. Laurenson, J. Hillston, Evaluation of RSVP and mobility-aware RSVP using performance evaluation process algebra, in: 2008 IEEE International Conference on Communications, 2008, pp. 192–197, <http://dx.doi.org/10.1109/ICC.2008.43>.
- [22] D. Jansen, Understanding Fox and Glynn's Computing Poisson Probabilities, CTIT Technical Report Series, 2011, URL <https://hdl.handle.net/2066/92528>.