Finding steady states of communicating Markov processes combining aggregation/disaggregation with tensor techniques

Francisco Macedo[1,2]

[1] EPF Lausanne, SB-MATHICSE-ANCHP, Station 8, CH-1015 Lausanne, Switzerland, francisco.macedo@epfl.ch
[2] IST, Alameda Campus, Av. Rovisco Pais, 1, 1049-001 Lisbon, Portugal

Abstract. Stochastic models for interacting processes feature a dimensionality that grows exponentially with the number of processes. This state space explosion severely impairs the use of standard methods for the numerical analysis of such Markov chains. In this work, we develop algorithms for the approximation of steady states of structured Markov chains that consider tensor train decompositions, combined with well-established techniques for this problem – aggregation/disaggregation techniques. Numerical experiments demonstrate that the newly proposed algorithms are efficient on the determination of the steady state of a representative set of models.

1 Introduction

We consider computing the stationary distribution of a continuous–time Markov chain, i.e., solving

$$Ax = 0 \text{ with } \mathbf{1}^T x = 1, \tag{1}$$

where A is the transpose of the generator matrix of the Markov chain and $\mathbf{1}$ denotes the vector of all ones. Matrix A is non-symmetric, singular and verifies $\mathbf{1}^T A = 0$.

We focus on Markov processes that feature high-dimensional state spaces arising from modelling subsystems that interact with each other. The considered Markov chain consists of d interacting subsystems (processes), and A consequently has a tensor (Kronecker) structure of the form

$$A = \sum_{t=1}^{T} \bigotimes_{k=1}^{d} E_k^t, \tag{2}$$

where each term in the summation represents a possible transition between states. Problems of this kind are known for their so called state space explosion [6] – exponential growth of the state space dimension on the number of subsystems – which severely impairs the numerical analysis of such Markov processes.

Applications of these models can be found, e.g., in queueing theory [7, 16]; stochastic automata networks [19, 26]; analysis of chemical reaction networks [1, 20]; or telecommunications [2, 25].

Recent algorithms that explore the Kronecker structure in (2) can be found in [3, 4, 18], where the first is a combination of the other two that intends to avoid the drawbacks of each. They use tensor techniques to solve (1) by using a compressed representation of matrices and vectors to avoid the enormous storage complexity corresponding to the large problem sizes and also to efficiently perform elementary operations, e.g. matrix-vector products. Vectors are, furthermore, approximated. This way, they deal with the state space explosion. The methods mentioned above use tensor train (TT) format, while canonical decomposition (CP) [17] had initially been used in [5]. However, the approximation of vectors inside the algorithms is fundamental and this format does not allow efficient approximations (truncations). This is not the case for TT format as the truncation procedure – TT-SVD algorithm [23] – is based on singular value decomposition (SVD), which is known to have quasi-optimal upper bounds for the error that is done in an approximation.

Aggregation/disaggregation techniques, which can be seen as a subclass of multigrid methods, have also been used to solve (1). A reduced version of (1) where states have been aggregated in groups is solved and then a solution for the original problem needs to be extrapolated using some disaggregation scheme. Aggregation can be defined, for instance: assuming an underlying Kronecker structure of the generator matrix [6, 8, 15, 27]; or imposing specific relations between the rates of transition between states, entries of the generator matrices, of the original and the aggregated processes, in a process called lumpability [9, 14].

The approaches proposed in this paper consist of using techniques from the aggregation/disaggregation class that are convenient to adapt to TT format. Two types of operators for aggregation, which result in two proposed algorithms, are considered. Our contribution consists of targeting the computation of the stationary distribution for finite-dimensional communicating Markov processes, developing and comparing different algorithmic approaches, testing their sensitivity to different models by using a broad benchmark collection.

The remainder of this paper is organized as follows. In Sec. 2 we briefly describe tensor-train format. An overview of the ideas behind aggregation/disaggregation algorithms, seen as a subclass of multigrid algorithms, followed by exploring the two mentioned concretizations of interest, is done in Sec. 3. The proposed algorithms, based on combining ideas from the two previous sections, are presented in Sec. 4. In Sec. 5, we analyse the performance of the proposed algorithms for some popular and representative models. Concluding remarks are given in Sec. 6.

2 Tensor train (TT) format

We recall the tensor train (TT) format for a vector of length $n_1 n_2 \cdots n_d$, when regarded as an $n_1 \times n_2 \times \cdots \times n_d$ tensor; and of the operator TT format that represents a matrix of sizes $n_1 n_2 \cdots n_d$.

2.1 Representation of a vector

For a network of d communicating processes, the steady state vector has length $n_1 n_2 \cdots n_d$, where n_μ denotes the number of states in the μ-th process, for $\mu = 1, \ldots, d$. Its entries can be naturally rearranged into an $n_1 \times \cdots \times n_d$ array, defining a d-th order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, whose entries are denoted by

$$\mathcal{X}_{i_1, i_2, \ldots, i_d}, \quad 1 \leq i_\mu \leq n_\mu, \quad \mu = 1, \ldots, d.$$

The opposite operation, typically called vectorization, denoted by $\mathrm{vec}(\mathcal{X})$, stacks the entries of $\mathcal{X}$ back into a long vector. The multi-indices of the form $(i_1, i_2, ..., i_d)$ are assumed to be traversed in *reverse lexicographical order*. The different n_μ, $\mu = 1, \ldots, d$, are typically called *mode sizes*.

$\mathcal{X}$ becomes a matrix for $d = 2$ and the definition of rank is unique. This rank can be computed using the singular value decomposition (SVD) [12]. The extension of this concept to $d > 2$ is not unique, and thus different notions of low rank decompositions for tensors have been developed – see [17] for an overview.

Tensor train (TT) decomposition [24] benefits from the locality of interactions. A tensor is represented as

$$\mathcal{X}_{i_1, \ldots, i_d} = G_1(i_1) \cdot G_2(i_2) \cdots G_d(i_d), \quad G_\mu(i_\mu) \in \mathbb{R}^{r_{\mu-1} \times r_\mu}, \tag{3}$$

where each $G_\mu(i_\mu)$ is a matrix of size $r_{\mu-1} \times r_\mu$ for $1 \leq i_\mu \leq n_\mu$. By definition, $r_0 = r_d = 1$. The third order tensors $\mathbf{G}_\mu \in \mathbb{R}^{r_{\mu-1} \times n_\mu \times r_\mu}$, whose slides are the matrices $G_\mu(i_\mu)$, form the building blocks of the TT format and are called the *TT cores*. One expects any explicitly given vector to be possible to write in the form (3), for sufficiently large values r_μ, using the TT-SVD algorithm [23].

Remark 1. Tensor train decomposition is expected to be more suitable for networks with an underlying topology of the processes associated with a train, in the sense that the existing interactions between subsystems should concern consecutive subsystems after they are suitably ordered; recall (3).

If the entries of the tuple $(r_1, r_2, ..., r_{d-1})$, called *TT rank*, remain modest, the complexity of storing and performing operations will be significantly reduced. For structured models (networks), its entries are expected to remain small.

2.2 Representation of a matrix

To efficiently apply a matrix to a vector in TT format, it also needs to be represented in a suitable way. We represent a matrix $A \in \mathbb{R}^{n_1 n_2 \cdots n_d \times n_1 n_2 \cdots n_d}$ using the *operator TT decomposition*

$$A_{(i_1, \ldots, i_d), (j_1, \ldots, j_d)} = A_1(i_1, j_1) \cdot A_2(i_2, j_2) \cdots A_d(i_d, j_d), \tag{4}$$

where $A_\mu(i_\mu, j_\mu) \in \mathbb{R}^{t_{\mu-1} \times t_\mu}$, $t_0 = t_d = 1$ and $(t_1, t_2, ..., t_{d-1})$ is the *TT rank*. The tensor $\tilde{\mathcal{X}}$ that results from a matrix-vector product with A, $\text{vec}(\tilde{\mathcal{X}}) = A \cdot \text{vec}(\mathcal{X})$, has a simple TT decomposition. Its new cores, $\tilde{G}_\mu$, are given by

$$\tilde{G}_\mu(i_\mu) = \sum_{j_\mu=1}^{n_\mu} A_\mu(j_\mu, i_\mu) \otimes G_\mu(j_\mu) \in \mathbb{R}^{r_{\mu-1} t_{\mu-1} \times r_\mu t_\mu}, \tag{5}$$

$i_\mu = 1, \ldots, n_\mu$. The entries of the TT rank thus multiply.

3 Aggregation/disaggregation schemes

We describe multigrid, as aggregation/disaggregation algorithms are a subclass of them, and then describe particular choices of the restriction (aggregation) operators that should be effective for Markov chains characterized by interacting subsystems.

3.1 Multigrid

Multigrid methods [13] use a set of recursively coarsened representations of the original setting to achieve accelerated convergence.

We describe the core multigrid V-cycle in Alg. 1.

The main ingredients that characterize this class of algorithms are: the smoothing scheme; the set of coarse variables; the transfer operators (restriction and interpolation operators); the coarse grid operator.

Algorithm 1: Multigrid V-cycle

$v_\ell = \text{MG}(b_\ell, v_\ell)$
if *coarsest grid is reached* **then**
  solve coarse grid equation $A_\ell v_\ell = b_\ell$.
else
  Perform ν_1 smoothing steps for $A_\ell v_\ell = b_\ell$ with initial guess v_ℓ
  Compute the residual $r_\ell = b_\ell - A_\ell v_\ell$
  Restrict $b_{\ell+1} = Q_\ell r_l$
  Restrict $A_{\ell+1} = Q_\ell A_\ell P_\ell$
  $v_{\ell+1} = 0$
  $e_{\ell+1} = \text{MG}(b_{\ell+1}, v_{\ell+1})$
  Interpolate $e_\ell = P_\ell e_{\ell+1}$
  $v_\ell = v_\ell + e_\ell$
  Perform ν_2 smoothing steps for $A_\ell v_\ell = b_\ell$ with initial guess v_ℓ
end

A more detailed description can be found in [30].

Summing up the main ideas in Alg. 1 in words: on the way down, at level ℓ, the method performs a certain number ν_1 of smoothing steps, using an iterative solver; the residual of the current iterate is computed and restricted by a matrix-vector multiplication with the restriction matrix for level ℓ, $Q_\ell \in \mathbb{R}^{m_\ell \times m_{\ell+1}}$, where m_ℓ is the number of states at level ℓ; the operator A_ℓ is also restricted via Petrov-Galerkin to get $A_{\ell+1} = Q_\ell A_\ell P_\ell, Q_\ell A_\ell P_\ell \in \mathbb{R}^{m_{\ell+1} \times m_{\ell+1}}$, where $P_\ell \in \mathbb{R}^{m_{\ell+1} \times m_\ell}$ is the interpolation operator at level ℓ; then we have a recursive call where we solve the coarse grid equation, which is the residual equation, in the last level; then, on the way up the grids, the error is interpolated and again some smoothing iterations are applied.

3.2 Restriction and interpolation on aggregation/disaggregation methods

What separates aggregation/disaggregation techniques [29] from general multigrid algorithms is that restriction is done in a particular way, associated with aggregation (while interpolation is then associated with disaggregation). A partition of the set of variables is defined and each set of this partition is then associated with one coarse variable.

From the proposed ways to aggregate the states, we focus on two variants in which aggregation (restriction) has a simple Kronecker representation. This is the case when assuming existence of subsystems inside the network.

Tensorized algorithm pairing states in each subsystem. Aggregation can be done in a tensorized way as proposed in the numerical experiments of [15]. In each subsystem, the states are merged in pairs. This method should be thus particularly suitable for networks for which the states are ordered in each subsystem, associated with a 1D topology.

The simple Kronecker representation of aggregation is strongly related with the fact that aggregation is tensorized. In fact, the associated matrix is a simple Kronecker product of smaller matrices. Each small matrix is associated with one subsystem. Such matrix is, for an example where the number of states of the subsystem of interest is 4,

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T . \tag{6}$$

In particular, defining m_ℓ as the number of states at level ℓ, $m_{\ell+1} = m_\ell / 2^d$ as the number of states per subsystem is divided by 2.

The way states are aggregated is represented in Fig. 1, for an example with $d = 2$ and again assuming 4 states per subsystem.

The suitability of a setting with an underlying ordering of the states in each subsystem, associated with a 1D topology, is clear in Fig. 1.

Aggregating all states from fixed subsystems. Aggregation can be also done, again assuming the existence of interacting subsystems, through aggregating all states of one particular subsystem in each level [6, 8].

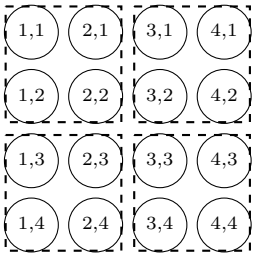

Fig. 1. Example of how aggregation works for $d = 2$ and 4 possible states per subsystem.

Aggregation is again represented with a simple Kronecker product of smaller matrices. Such matrices are now all identity except the one for the dimension associated with the subsystem whose states are aggregated. For that particular dimension, we have a column vector of ones. In particular, $m_{\ell+1} = m_\ell / n_i$, where i is the dimension associated with the subsystem whose states are aggregated from level ℓ to $\ell + 1$.

The way states are aggregated is represented in Fig. 2, again for an example with $d = 2$ and 4 states per subsystem.

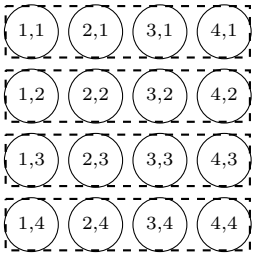

Fig. 2. Example of how aggregation works for $d = 2$ and 4 possible states per subsystem, assuming that the first dimension is the one associated with the subsystem whose states are aggregated.

Two states are aggregated from one level to the next if they are in the same states in a subset of the subsystems that includes all except one, the one whose states are aggregated. In particular, any local topology is now ignored. In Fig. 2, the subsystem whose state is not relevant for the aggregation is the first.

Remark 2. Figure 2 clearly shows how, when the mode sizes start to increase, the distance between states that are aggregated together increases, in case the local topology is associated with ordered states – 1D topology.

4 Proposed algorithms

In terms of the transpose of the generator matrix of the Markov chain A, the computation of the steady state requires the solution of (1). We focus on problems where the matrix A has the form (2).

4.1 First variant of the algorithm from Sec. 3.2 in TT format

Our first proposed algorithm combines the first variant of the algorithm from Sec. 3.2 with TT format by having all involved structures, in particular matrices and vectors, in this format.

Restriction and interpolation. As the corresponding aggregation is represented as a simple Kronecker product of small matrices, as seen in Sec. 3.2, it allows a TT representation with all entries of the TT rank equal to 1. The relationship between having a simple Kronecker representation and having a simple TT representation is clear. The small matrices from the Kronecker representation are A_μ, $\mu = 1, ..., d$, in the TT representation (4). Such matrices are of the form (6) (adapted to the mode sizes).

We use the transpose of restriction for interpolation, which again has a TT representation where the entries of the TT rank are all 1. In fact, we just need to transpose each of the cores (matrices) from the TT representation of restriction.

Smoother. The tensor structure of the problem rules out smoothers that require access to individual entries of the operator, e.g., Jacobi or Gauss-Seidel. Good candidates are smoothers that only require operations that can be cheaply done within the format. Krylov subspace methods are thus good candidates. We use GMRES, following [4].

Coarse grid solver. Coarse grid is still affected by curse of dimensionality as the mode sizes get reduced from one level to another but not the number of dimensions. This is the same type of tensorized multigrid scheme that is used in the scheme proposed in [4], which in particular also suffers from this problem. For that case, a solution, which we adopt in this algorithm, was proposed in [3]. The idea is to use AMEn [10, 11] as coarse grid solver.

Implementation details.

Smoother. For smoothing, we use three steps of GMRES in the finest grid while one step in the remaining grids.

Normalization. In (1) we have the restriction that the sum of the entries of the solution is 1. This is not naturally kept during a cycle. We thus normalize the obtained approximation after each cycle.

Truncations. Truncation is needed during a cycle to prevent excessive rank growth. We use the already mentioned TT-SVD algorithm. Looking at Alg. 1, the TT rank increases in all steps, including inside the smoother, except when applying interpolation and restriction. Truncation is performed after each of these steps.

r_ℓ is truncated with constant accuracy 10^{-1}; the truncation of v_ℓ uses an adaptive scheme where the target accuracy depends on the residual after the previous cycle, being this value times a constant, 10; the accuracy of the truncation of v_ℓ is also used for the truncations inside the GMRES smoother.

A restriction on the maximum rank that one can obtain after each truncation is needed to avoid excessive rank growth. This restriction, originally proposed in [4], depends on the cycle, starting with value 15 and increasing by a factor of $\sqrt{2}$ after cycles for which the variation of the residual norm is smaller than a factor of 0.9 to avoid stagnation.

Size of the coarse grid problem. By construction of restriction and interpolation, the mode sizes in the coarse grid can only be powers of 2. Tests show that mode sizes 4 are already too large so that we consider mode sizes 2.

4.2 Second variant of the algorithm from Sec. 3.2 in TT format

The second proposed algorithm combines the second variant of the algorithm discussed in Sec. 3.2 with TT format by again considering the algorithm with all structures in this format.

The comments concerning restriction and interpolation, and also the smoothing procedure, are just the same as for the previous algorithm; see Sec. 4.1.

Coarse grid solver. While in the algorithm from Sec. 4.1 the aim of the multigrid scheme is to reduce the mode sizes from one level to another, maintaining the original value of d; in this one we maintain the mode sizes but reduce d. AMEn is thus not so suitable and it is replaced with Moore-Penrose pseudoinverse.

Implementation details. Most implementation details are as in the algorithm from Sec. 4.1.

Size of the coarse grid problem. As the expensive Moore-Penrose pseudoinverse is the coarse grid solver, we define the number of levels as the smallest value for which the number of states on the coarse grid is smaller than 350.

5 Numerical experiments

We analyse the performance of the algorithms proposed in Sec. 4. The algorithms of reference for comparison, given their effectiveness when compared against previously proposed algorithms and that they are also in TT format, are the algorithms proposed in [3] and [18]. They are called MultigridAMEn and AMEn,

respectively. As for the algorithm proposed in [4], it might be also considered but as it has the same core as MultigridAMEn, differing only in the coarse grid solver, where the one from MultigridAMEn is proposed to avoid the curse of dimensionality that affects the one from the algorithm of interest; as noted in the implementation details of Sec. 4.1; we do not consider it.

All tests were performed in MATLAB version 2013b, using functions from *TT-Toolbox* [22].

The algorithms from Sec. 4.1 and Sec. 4.2 are called 'TensorizedAggregation' and 'DimAggregation', respectively.

Throughout all experiments, we stop an iteration when the residual norm $\|Ax\|$ is two orders of magnitude smaller than the residual norm of the tensor of all ones (scaled so that the sum of its entries is one).

All computation times were obtained on a 12-core Intel Xeon CPU X5675, 3.07GHz with 192 GB RAM running 64-Bit Linux version 2.6.32.

We use n from here, in the sequence of Sec. 2.1, for the value that is considered for all the mode sizes of a given test case as we will assume equal model sizes in all dimensions.

In the tables that follow: 'Time' stands for the computation time, in seconds; 'Rmax' stands for the maximum entry of the TT rank of the approximated solution; 'Iter' stands for the number of iterations that is needed (measured in cycles for all algorithms except for AMEn, for which it is measured in sweeps, see [18]). We use '-' for algorithms that do not converge.

The experiments are done on some models contained in the collection [21], wide and representative collection of models associated with interacting subsystems. We vary n and d; and use, for the remaining input parameters, the natural generalizations of the default ones.

Each of the following subsections is associated with a different model. For more details, see [21], where the corresponding model is easily found as the names given to the models coincide.

5.1 Model convergingmetab in [21]

We consider a model [20] from the field of chemical networks which has a topology of interactions that should not suit TT format particularly well, in the sequence of Rem. 1.

We consider $n = 4$ and $d = 10$, respectively; see Tab. 1.

AMEn has an excellent performance. It is, in fact, only expected to have problems if the entries of the TT rank vector are large as this would imply a significant increase on the cost of the subproblem that must be repeatedly solved, as discussed in [18].

As for the proposed algorithms, we see that the restriction (aggregation) operator that is used in TensorizedAggregation is effective as it is the only part where it differs from MultigridAMEn.

DimAggregation behaves particularly well in this context with small mode sizes, which is expected as it is only expected to have problems when the mode sizes are large; recall Rem. 2.

Table 1. Comparison of the algorithms for model convergingmetab – $4^{10} \approx 1.05 \times 10^6$ states.

	Time	Rmax	Iter
MultigridAMEn	37.4	15	13
AMEn	1.8	16	4
TensorizedAggregation	31.0	29	13
DimAggregation	14.9	15	8

Globally, the algorithms, all in TT format, seem to be effective even when the underlying topology is not the theoretically most suitable one, associated with Rem. 1.

5.2 Model cyclemetab in [21]

We now consider another model [20] from the field of chemical networks, with a topology that is very far from the ideal one for TT format, see Rem. 1, as there is a cycle – last and first subsystems also interact. Additionally, this model is reducible, meaning in particular that the steady state is not unique. This is a class of models that algorithms for finding steady states of Markov chains tend to avoid.

Note that while the methods are intended to be applied to irreducible Markov chains, in the reducible case we have connected components on the set of states so that in order to obtain the desired probabilities from the obtained values, one only needs to then normalize them in a way that the probabilities associated with each connected component sum to 1, as, in the end, each connected component is associated with a different irreducible problem. In particular, the probabilities become unique again.

We go further on n and d, one at each time, considering two cases.

First case – $n = 4$ *and* $d = 18$. The results for the first case are in Tab. 2.

Table 2. Comparison of the algorithms for model cyclemetab – $4^{18} \approx 6.87 \times 10^{10}$ states.

	Time	Rmax	Iter
MultigridAMEn	525.5	80	22
AMEn	418.7	31	6
TensorizedAggregation	82.2	57	17
DimAggregation	80.9	41	10

The proposed algorithms perform globally well despite the mentioned complicated particularities of this model, considering that the large number of di-

mensions that is now considered leads to a problem size that is larger than the previous ones. The resulting large entries of the TT rank vector lead to a bad performance of AMEn, while MultigridAMEn is also clearly outperformed by the proposed methods.

The proposed algorithms might perform even better in case there was more control on the rank growth, noting that the entries of the TT rank vector could be much smaller by looking at the maximum entry obtained with AMEn. In fact, the restriction on the maximum entries of the TT rank after the different truncations that is imposed on these algorithms, which is described in the implementation details in Sec. 4.1, is not enough to avoid that the entries of the TT rank become too large. This limitation also affects MultigridAMEn.

The fact that the algorithms are effective even when the underlying topology is not suitable is emphasized as the topology from this model is, as noted before, extremely far from the ideal one.

Second case – $n = 32$ *and* $d = 6$. We address the second case in Tab. 3.

Table 3. Comparison of the algorithms for model cyclemetab – $32^6 \approx 1.07 \times 10^9$ states.

	Time	Rmax	Iter
MultigridAMEn	432.1	80	19
AMEn	1956.5	31	6
TensorizedAggregation	64.2	57	15
DimAggregation	-	-	-

The performance of DimAggregation is strongly affected by increasing the mode sizes, in the sequence of Rem. 2. As for AMEn, it is also more suited for problems with a large number of dimensions than for problems with large mode sizes, as the problem is more structured in the first case; recall Rem. 1; thus resulting in this particularly poor performance when the mode sizes are increased, even if the global problem size is smaller than in Tab. 2.

5.3 Model handoff2class1d in [21]

This model [2,28] – from the telecommunications field – has a topology of interactions that suits TT format well, in the context of Rem. 1. Its main particularity is the existence of two distinguishable types of customers.

As the 1D topology of each subsystem, associated with a natural ordering of the states, is lost, algorithms TensorizedAggregation and MultigridAMEn are not suitable.

Also because of the loss of the local 1D topology, Rem. 2 does not apply. The distance between states is not as straight-forward in the new underlying

topology but it is clear that it is not as in Fig. 2, and, furthermore, that the distance between states is now smaller. We thus expect DimAggregation to perform properly when n grows.

We go even further on the mode sizes and number of dimensions, considering again two cases.

First case – $n = 10$ *and* $d = 28$. The results for the first case are in Tab. 4.

Table 4. Comparison of the algorithms for model handoff2class1d – 10^{28} states.

	Time	Rmax	Iter
AMEn	4.7	4	3
DimAggregation	149.7	21	4

AMEn and DimAggregation can easily deal with 28 dimensions, which is expected as they are, as noted before, particularly suited for large d.

In the case of AMEn, as the entries of the TT rank vector are very small, this algorithm is very hard to beat.

The reason why the computation time is much worse for DimAggregation is the problem that was mentioned in the context of Tab. 2 concerning the far too large entries of the TT rank.

Note that the problem size that is being addressed is clearly the largest that was tested – total of 10^{28} states.

Second case – $n = 210$ *and* $d = 4$. We address the second case in Tab. 5.

Table 5. Comparison of the algorithms for model handoff2class1d – $210^4 \approx 1.94 \times 10^9$ states.

	Time	Rmax	Iter
AMEn	231.4	8	6
DimAggregation	134.7	29	5

Despite the smaller problem size, when compared with Tab. 4, AMEn behaves worse because of the larger mode sizes that are now considered; same argument as in the comparison of the performances of AMEn in the tables from Sec. 5.2.

We see that DimAggregation can deal with such a value of n. In fact, while mode sizes 32 are too large for the model in Sec. 5.2, recall Tab. 3, as expected in the sequence of Rem. 2, we now get good convergence for mode sizes 210 as the remark does not apply.

Despite not using any information about the (now more complex) topology of each subsystem, DimAggregation seems to perfectly address models with distinguishable customers. Note that it would be even more competitive if the adopted rank adaptivity scheme would not generate too large entries of its TT rank.

5.4 Choosing the algorithm to use

We now propose a way to decide which algorithm to use *depending on the type of model*.

AMEn would be the best option if it was not so strongly affected by cases in which the entries of the TT rank become large which is expected to hold in the most interesting case studies in practice as these should involve large problem sizes. In particular, problems with large mode sizes are common and this is clearly enough, as noted before, for AMEn to stop being efficient.

As for MultigridAMEn, while it is clearly a good option as it performs well for a large and representative collection of models as seen in [3], the restriction and interpolation that are chosen seem to be possible to outperform when replacing them with the aggregation and disaggregation operators proposed in TensorizedAggregation – algorithm from Sec. 4.1 – as repeatedly seen in the tables throughout this section.

In the end, AMEn should be dismissed while MultigridAMEn is outperformed by TensorizedAggregation. TensorizedAggregation and DimAggregation are thus the algorithms of reference.

Models with distinguishable customers have to be addressed using DimAggregation. This algorithm has a particularly good performance, even when the mode sizes increase given that the 1D topology of each subsystem is lost so that Rem. 2 is not a problem. As Rem. 2 applies to models with indistinguishable customers, DimAggregation should not be used for those. This is, however, a context that perfectly suits TensorizedAggregation. In fact, the proposed algorithms are somehow complementary. For each of the two classes in which we have just partitioned the set of models, there is thus one of the algorithms proposed in this paper that perfectly suits it.

6 Conclusion

We have proposed algorithms for approximating steady states for structured Markov chains combining the concepts of aggregation/disaggregation with TT decompositions.

Numerical experiments with stochastic automata networks demonstrate that, for general problem sizes, they can easily outperform two strong recent algorithms proposed in [18] and [3]. They can, furthermore, perform remarkably well for very high-dimensional problems. The largest problem size that is addressed is 10^{28}; the largest mode sizes are 210; the maximum number of dimensions is 28.

With the proposed algorithms, all models deriving from Markov chains with a simple Kronecker representation should be possible to address. Robustness is, in fact, one of their main strengths. Besides covering very different fields of interest (chemical networks, queueing networks and telecommunications), the tests done in this paper include: a reducible model, for which the performances of the algorithms, both in comparative and absolute terms, seem to be similar to the performances for a general irreducible model; a model with distinguishable customers, which should be particularly challenging but for which we have an algorithm that perfectly deals with it. These two classes of models are extremely important, both because of their range of applications; but also because of the difficulties, in the past, in addressing them. Additionally, models with topologies that are not suitable for applying TT format provide surprisingly good results.

In the context of the mentioned robustness, when choosing the algorithm to use, depending on the type of model, the choice is quite natural, as seen in Sec. 5.4.

One important limitation that these algorithms have is on the control of the rank growth. Even with a restriction on the maximum entries of the TT rank allowed after the different truncations, the entries of the TT ranks can grow far too much, clearly influencing their performance. This should be further investigated.

Acknowledgments. I thank Daniel Kressner (EPF Lausanne) for helpful discussions.

References

1. D. F. Anderson, G. Craciun, and Th. G. Kurtz. Product-form stationary distributions for deficiency zero chemical reaction networks. *Bull. Math. Biol.*, 72(8):1947–1970, 2010.
2. Nelson Antunes, Christine Fricker, Philippe Robert, and Danielle Tibi. Analysis of loss networks with routing. *Annals of Applied Probability*, 16(4):2007–2026, 2006.
3. M Bolten, K Kahl, D Kressner, F Macedo, and S Sokolović. Multigrid methods combined with amen for tensor structured Markov chains with low rank approximation. *arXiv preprint arXiv:1605.06246*, 2016.
4. Matthias Bolten, Karsten Kahl, and Sonja Sokolović. Multigrid methods for tensor structured Markov chains with low rank approximation. *arXiv preprint arXiv:1412.0937*, 2014.
5. P. Buchholz. Product form approximations for communicating Markov processes. *Perform. Eval.*, 67(9):797–815, 2010.
6. P. Buchholz and T. Dayar. On the convergence of a class of multilevel methods for large sparse Markov chains. *SIAM J. Matrix Anal. Appl.*, 29(3):1025–1049, 2007.
7. R. Chan. Iterative methods for overflow queueing networks I. *Numer. Math.*, 51:143–180, 1987.
8. Tugrul Dayar. *Analyzing Markov chains using Kronecker products: theory and applications.* Springer Science & Business Media, 2012.
9. Salem Derisavi, Holger Hermanns, and William H Sanders. Optimal state-space lumping in markov chains. *Information Processing Letters*, 87(6):309–315, 2003.

10. S. V. Dolgov and D. V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. part I: SPD systems. *ArXiv e-prints*, January 2013.
11. S. V. Dolgov and D. V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. part II: Faster algorithm and application to nonsymmetric systems. *ArXiv e-prints*, April 2013.
12. G. H. Golub and Ch. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.
13. W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, 2003.
14. Jane Hillston, Andrea Marin, Sabina Rossi, and Carla Piazza. Contextual lumpability. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, pages 194–203. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.
15. G. Horton and S. T. Leutenegger. A multi-level solution algorithm for steady-state Markov chains. In B. D. Gaither, editor, *Proceedings of the ACM SIGMETRICS 1994 Conference on Measurement and Modeling of Computer Systems*, pages 191–200, 1994.
16. L. Kaufman. Matrix methods for queuing problems. *SIAM J. Sci. Statist. Comput.*, 4:525–552, 1983.
17. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.
18. D. Kressner and F. Macedo. Low-rank tensor methods for communicating Markov processes. In Gethin Norman and William Sanders, editors, *Quantitative Evaluation of Systems*, volume 8657 of *Lecture Notes in Computer Science*, pages 25–40. Springer International Publishing, 2014.
19. A. N. Langville and W. J. Stewart. The Kronecker product and stochastic automata networks. *J. Comput. Appl. Math.*, 167(2):429 – 447, 2004.
20. E. Levine and T. Hwa. Stochastic fluctuations in metabolic pathways. *Proc. Natl. Acad. Sci. U.S.A.*, 104(22):9224–9229, 2007.
21. F. Macedo. Benchmark problems on stochastic automata networks in tensor train format. Technical report, MATHICSE, EPF Lausanne, Switzerland, 2015.
22. I. V. Oseledets. MATLAB TT-Toolbox Version 2.2, 2011. Available at `http://spring.inm.ras.ru/osel/?page_id=24`.
23. I. V. Oseledets. Tensor-Train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.
24. Ivan V Oseledets and Eugene E Tyrtyshnikov. Breaking the curse of dimensionality, or how to use svd in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009.
25. Bernard Philippe, Youcef Saad, and William J. Stewart. Numerical methods in Markov chain modelling. *Operations Research*, 40:1156–1179, 1996.
26. B. Plateau and W. J. Stewart. Stochastic automata networks. In *Computational Probability*, pages 113–152. Kluwer Academic Press, 1997.
27. Ivana Pultarová and Ivo Marek. Convergence of multi-level iterative aggregation–disaggregation methods. *Journal of computational and applied mathematics*, 236(3):354–363, 2011.
28. Moshe Sidi and David Starobinski. New call blocking versus handoff blocking in cellular networks. *Wirel. Netw.*, 3(1):15–27, March 1997.
29. W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.
30. U. Trottenberg, C. Osterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.