Springer Series in Operations Research
and Financial Engineering

Tuğrul Dayar

Kronecker Modeling and Analysis of Multidimensional Markovian Systems

Springer

Springer Series in Operations Research and Financial Engineering

More information about this series at http://www.springer.com/series/3182

Tuğrul Dayar

Kronecker Modeling and Analysis of Multidimensional Markovian Systems

Springer

Tuğrul Dayar
Department of Computer Engineering
Bilkent University
Ankara, Turkey

ISSN 1431-8598 ISSN 2197-1773 (electronic)
Springer Series in Operations Research and Financial Engineering
ISBN 978-3-319-97128-5 ISBN 978-3-319-97129-2 (eBook)
https://doi.org/10.1007/978-3-319-97129-2

Library of Congress Control Number: 2018951794

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To all those from whom we have learned something

Preface

This book is about the Markovian modeling and analysis of systems with a discrete multidimensional state space in which various transitions take place. The models of interest are composed of submodels, each dimension corresponds to a submodel that represents a different subsystem, and submodels interact through possible transitions. Particular emphasis is on multidimensional models having unreachable states that appear in the product state space of submodels but can never be occupied in practice. The book shows step-by-step how the transition matrix of the model associated with the reachable state space can be represented compactly and exactly on computer using sums of Kronecker products of matrices associated with submodels. Then the analysis amounts to the numerical computation of steady-state or transient solutions of very large such models as fast, as accurate, and in as little of memory as possible.

Numerical analysis suggests a group of algorithmic methods that are to be implemented on computer and yield solutions to mathematical problems with errors that can be assessed and controlled so that the solutions have accuracies relatively close to the available finite computer precision. Since trying to compute a solution more accurate than computer precision is futile in practice, the concept of a solution being exact to computer precision is used. This is about 16 decimal digits in double precision IEEE 754 standard. In other words, every solution that is obtained on computer can be qualified as being approximative unless the first 16 decimal digits of the solution obtained on computer is equal to the first 16 decimal digits of the solution obtained with paper and pencil. Nevertheless, the term approximative is generally not used when referring to a numerical method. The accuracy of the computed solution, which is a probability vector, is rather quantified by indicating its residual or error norm.

The material in this book has grown out of work spanning the last 20 years of our research after receiving the PhD degree and is based on our earlier Briefs book from 2012 [101], which is of 86 pages excluding its preamble.

The Briefs book is primarily about analysis as can be inferred from its title and does not discuss unreachable states except for a few pages in its Section 4.2. To make the exposition more accessible, the Briefs book motivates the discussion with a model having no unreachable states. The particular model which is first introduced in (1.1) on page 3 through its (12×12) transition matrix corresponds to a model with 3 submodels, respectively, having 2, 3, and 2 states, hence the $2.3.2 = 12$ states in the state space. This is a relatively small model that still has meaningful semantics as described on page 4. The reason behind choosing 3 rather than 2 submodels is simply that in mathematics many times one does not need to go beyond 3 to see a common pattern and start making generalizations, but results obtained with 2 may be misleading. Furthermore, we could only fit the (16×16) transition matrix of a model with 4 submodels to a page only if all submodels had 2 states, and that would imply too much symmetry in the semantics of the submodels. Hence, we chose 3 submodels and made one of them have 3 states and the other two 2 states. The Kronecker representation in Chapter 2 is discussed in terms of this model. All iterative methods in Chapter 3 and the decompositional iterative method on pages 39–41 in Section 4.1 are based on this model as well. The only exception is the case study in Section 4.3 related to closed queueing networks that have unreachable states. For this closed queueing network, the considered solution method is termed approximative since the decomposition of the closed queueing network into subnetworks is accomplished through first-order measures and the results it provides are rather crude, being accurate only to a few decimal digits. Hence, a lot of thought has gone into the selection of the running example which does not have unreachable states and the organization of the Briefs book around that example.

The organization of the current book and its contents including the examples are significantly different from those of the Briefs book. The current book is of 269 pages excluding its preamble. Its first part in Chapter 2 through Chapter 4 discusses multidimensional Markovian modeling with unreachable states using Kronecker products, and its second part in Chapter 5 through Chapter 7 discusses analysis. Its study requires one to understand how unreachable states are avoided through a Cartesian product partitioning of the reachable state space. Noticing that we are able to fit an (18×18) transition matrix to a page, the running example in the Briefs book has been changed in this book to correspond to a model with 3 submodels, respectively, having 3, 2, and 3 states, hence the $3.2.3 = 18$ states in the state space. This example is introduced in (1.1) on page 4 with the corresponding figure on page 6. The transition matrix of this example seems to be the largest we can fit legibly to a page. This example, which does not have any unreachable states, is later used on pages 17–19 in Chapter 2 to motivate stochastic automata networks (SANs) and on pages 20–21 to illustrate the nested block partitionings associated with the transition matrix. On pages 21–24 in Chapter 2, a second small running example which also has 18 states is given with its figure to explain the concepts of unreachable states and functional transitions in SANs. This

second running example is used on pages 26–29 to define the reachable state space consisting of 12 states as a union of Cartesian products and develop the block representation of its (12×12) transition matrix in which each block is a sum of Kronecker products. Unfortunately, this cannot be done with the SAN approach. We believe these two examples to be small enough, yet meaningful, with transition matrices that fit legibly to a single page and can be followed by the reader. Chapter 2 continues with two other multidimensional examples coming from the areas of systems availability and queueing networks. These are followed by two other examples from the areas of production lines and communication protocols in Chapter 3 and three other examples from the areas of systems of stochastic chemical kinetics and queueing networks in Chapter 4. There is an eighth multidimensional example model from systems of stochastic chemical kinetics that appears in Chapter 7. In particular, the four example models in Chapter 4 and Chapter 7 have countably infinite state spaces. Some readers may find some of these multidimensional examples very detailed and repetitive. However, each example demonstrates a different aspect of the modeling process, and for instructional purposes we prefer to be precise and complete rather than to gloss over the details. In the second part of the book, a more efficient vector-Kronecker product multiplication algorithm is given and recent results regarding the compact representation of solution vectors with some controllable approximation error are discussed in Chapter 5. All iterative numerical methods in Chapter 6 are presented for models that may have unreachable states. Furthermore, now there is a separate chapter (i.e., Chapter 7) on transient analysis. As such, this book contains substantial new material and can be considered to be at a higher level than the Briefs book.

The style and language of each book reflects the experience, view, and taste of its author(s). So does this one. The subject matter in its broadest setting can be considered to be in the area of performance modeling and analysis. This is an area which is generally not taught in undergraduate computer science and engineering programs. If there is a related undergraduate course available in a university program, it is almost always elective. Some of the modeling formalisms used in this area, such as queueing networks, are also part of another area called operations research. However, operations research is also an area that mostly exists at the graduate level. Hence, our intention was not to write an undergraduate textbook on the Kronecker modeling and analysis of multidimensional Markovian systems. The book is clearly for graduate students and researchers. Furthermore, the analysis methods it advocates are all numerical.

We are fortunate to have taken two required courses on numerical analysis during our junior year in computer engineering at Middle East Technical University. Nowadays, most undergraduate programs in computer science and engineering do not have even a single required course on numerical analysis in their curriculum. This is a pity since, in our view, numerical computation can only be studied if one has taken at least one course in that direction after two

courses on calculus, a course on linear algebra, and a course on differential equations, all of which we are happy to have taken during our undergraduate studies. In addition to other required mathematics-related courses during our computer engineering studies such as mathematical logic, discrete mathematics, and probability and statistics, and other pertinent required courses such as data structures, programming languages, file organization and processing, digital computer fundamentals, computer organization, microprocessors, data management, database management systems, operating systems, systems programming, language processors, software engineering, systems concepts and modeling, formal languages and abstract machines, operations research techniques, data communications, and system simulation, we are also happy to have taken during our MS and PhD studies in computer science at North Carolina State University the courses of algorithms and automata theory with Matt Stallmann, operating systems with Bob Fornaro, parallel computer architecture with Ed Davis, numerical analysis (which we first audited with Bob White) with the late Bob Funderlic, matrix theory and applications with the late Gil Koh, stochastic processes with the late Oscar Wesler, queueing stochastic service systems with Russell King, computer communications with Wushow Chou, digital communications with Alexandra Duel–Hallen, performance evaluation with Arne Nilsson, data networks and computer network performance evaluation with Yannis Viniotis (our MS thesis advisor), and advanced performance evaluation with Billy Stewart (our PhD thesis advisor).

We must thank Yannis Viniotis and Harry Perros, our MS and PhD advisory committee members, for the queueing systems seminars they held and their very quantitative approaches to problem solving that have helped to advance our research. It was through our PhD thesis advisor Billy Stewart, our PhD thesis advisory committee member Carl Meyer, and Bob Funderlic that we came to appreciate the importance of floating-point representation and arithmetic. In our opinion, one of the best treatments of this subject matter appears in the first two chapters of [185] by Nick Higham. We have been using these chapters from its first edition in our undergraduate numerical analysis course since the second half of 1990s together with the matrix–vector approach to scientific computing discussed by Charles Van Loan in [319], and recommend them as a starting point to all who work in numerical computing. We are also fortunate to have met Pete Stewart at the Cornelius Lanczos Centenary Conference in Raleigh at the end of 1993 and have learned from his work that sometimes a (2×2) matrix is not large enough to illustrate a concept or notice a pattern associated with matrices, one needs a (3×3) matrix, and almost always this slightly larger matrix is sufficient and does the job. Another important figure has been Yousef Saad, whom we met for the first time at the 1994 Copper Mountain Conference on Iterative Methods. We followed the first edition of his book [280] in our introductory graduate level course "Iterative Methods for Sparse Linear Systems," and his work has

been influential in putting our view of iterative methods and preconditioners into perspective over the years.

Billy Stewart and Brigitte Plateau introduced us to SANs during our visit to Grenoble in June 1996. A study on robotic tape libraries using SANs was carried out in August that year with Odysseas Pentakalos and Brooke Stephens in Greenbelt, Maryland. The help received during this process from Paulo Fernandes regarding the software was instrumental. Later we had many interesting talks with Jean-Michel Fourneau and Franck Quessette over SANs in Versailles and Ankara from 1998 to 2000. Nihal Pekergin was also present during these visits and brought her expertise on stochastic comparison into the picture. In June 2000, we had enjoyable discussions with the late Ivo Marek and Petr Mayer in Prague on the convergence properties of iterative aggregation–disaggregation.

In the 2002–2003 academic year, we got a chance to learn about hierarchical Markovian models (HMMs) from Peter Buchholz in Dresden. Our discussions on HMMs continued in Dortmund in June 2005 and at Dagstuhl in February 2007. His habilitation thesis [46] includes work along the line of research discussed in this book. It was in Dortmund and then Ankara where we investigated compositional Markovian models for symmetries with Peter Kemper in 2005. Kishor Trivedi visited Ankara in 1997; he was always available by e-mail in 2008 while we were writing a joint paper and was ready for further discussions later that year in Seattle.

In early 2010 in Saarbrücken, we were convinced by Verena Wolf of the difficulties associated with analyzing systems of stochastic chemical kinetics using SANs and HMMs. This also led to stimulating exchanges during the same year with Holger Hermanns, Werner Sandmann, David Spieler, and later in March 2012 in Kaiserslautern and Saarbrücken with Hendrik Baumann. Udo Krieger, whom we met for the first time in 1999 in Zaragoza and saw on a number of different occasions later, has always been illuminating with his broad knowledge. It was also through him that we became aware of the meeting in June 2012 in Varese where we had insightful discussions with Dario Bini, Guy Latouche, Beatrice Meini, and Peter Taylor on matrix analytic methods. It was through Jeffrey Hunter that we were introduced to a part of the statistics community. We continued learning from one or more of these colleagues on different occasions in Atlanta in October 2015, Madeira and Budapest in June 2016, and Pisa in February 2017. It was due to Miklós Telek and András Horváth that we visited Budapest for the first time in September 2009 before our first visit to Pisa in October 2009. It was also our revisit to Prague in June 2013 through Ivana Pultarová's work that enabled us to catch up with the recent advances in multilevel iterative methods.

After Peter Buchholz's visit to Bilkent in August 2014, we visited his group at the Technical University of Dortmund in January 2016 to initiate a collaboration on compact solution vectors using the hierarchical Tucker decomposition for the solution of Markov chains based on Kronecker products. This book has been finalized thereat during our stay in 2017 with the

Modeling and Simulation group to which we owe special words of gratitude, especially to Falko Bause for his to-the-point technical expertise, and to Iryna Dohndorf and Jan Kriege for the collegial environment they have provided and their discussions.

We thank all the people above and those whose names we have not been able to list for providing a scholarly atmosphere in which to carry out research and for their time over the years. We met Erol Gelenbe in Raleigh while studying for the PhD degree, organized two international conferences with him and our colleagues at Bilkent in October 1998 and 2004 in Antalya, and visited him in London in September 2015, with other encounters in between. Our interaction with him has helped us to better organize and develop a more holistic view toward research. He will always be remembered. It was through Michele Benzi and Daniel Szyld in the 1990s that we started to learn many different aspects of numerical linear algebra. We have been able to keep in touch over the years, and along Billy Stewart, Peter Buchholz, Jean-Michel Fourneau, and Brigitte Plateau, they are probably the two most influential people on our research, deserving a special thank you.

We are grateful to Bilkent for being very understanding and generous in granting us research leaves, without which it would not have been possible to complete this book. We are also fortunate to have worked with a number of students at Bilkent who found Kronecker structured compositional Markovian models such as SANs and HMMs interesting and exciting: Ertuğrul Uysal, Oleg Gusak, Akın Meriç, İlker Nadi Bozkurt, and M. Can Orhan.

Grants available in one form or another over the years allowed us to continue working on the subject from the Turkish Scientific and Technological Research Council, the French National Scientific Research Center, the Center of Excellence in Space Data and Information Systems, the Czech Technical University, the Alexander von Humboldt Foundation, the Turkish Academy of Sciences, the Cluster of Excellence in Multimodal Computing and Interaction, and the Technical University of Dortmund.

Finally, we thank Donna Chernyk and her team at Springer for initiating the project following our earlier book [101] with them, steering us through the process, and making it happen. Special thanks are due to Agnes Felema. A from SPI Global for her effort during the production of the book. We also thank the anonymous reviewers who have enabled us to improve the presentation. We hope you enjoy the outcome.

Bilkent, Ankara, Turkey Tuğrul Dayar
June 2018

Contents

List of Figures

List of Tables

List of Symbols

Scalars

α_k	Rate of transition k for $k = 1, \ldots, K$
$\alpha_k^{(h)}$	Submodel h's state-dependent rate of transition k for $h = 1, \ldots, H$, $k = 1, \ldots, K$
b_l	Number of diagonal blocks at level l for $l = 0, \ldots, H$
H	Number of submodels (or dimensions) in multidimensional Markov chain, $H = H_I + H_F$
H_F	Number of submodels (or dimensions) with finite state spaces in multidimensional Markov chain
H_I	Number of submodels (or dimensions) with countably infinite state spaces in multidimensional Markov chain
K	Number of transitions, $K = \lvert\mathcal{K}\rvert$
n	Product state space size, $n = \lvert\mathcal{S}\rvert = \prod_{h=1}^{H} n_h$
n_h	State space size of submodel h, $n_h = \lvert\mathcal{S}^{(h)}\rvert$, for $h = 1, \ldots, H$
nz	Number of nonzeros in off-diagonal part of Q
$nz_{Q_k^{(h)}}$	Number of nonzeros in $Q_k^{(h)}$
N	Number of reachable state space partitions
$N_{\min}$	Minimum number of reachable state space partitions
o_l	Order of diagonal blocks at level l for $l = 0, \ldots, H$
ϕ_k	State-independent rate of transition k for $k = 1, \ldots, K$

Sets

$\mathcal{K}$	Set of transitions, $\mathcal{K} = \bigcup_{p,w=0}^{N-1} \mathcal{K}_{p,w}$
$\mathcal{K}^{(h)}$	Set of transitions in which submodel h participates for $h = 1, \ldots, H$
$\mathcal{K}_{p,w}$	Set of transitions associated with $Q(p, w)$ for $p, w = 0, \ldots, N-1$
$\mathcal{R}$	Reachable state space, $\mathcal{R} \subseteq \mathcal{S}$
$\mathcal{R}_p$	Reachable state space partition p for $p = 0, \ldots, N-1$, $\mathcal{R} = \bigcup_{p=0}^{N-1} \mathcal{R}_p$, $\mathcal{R}_p \cap \mathcal{R}_w = \varnothing$ for $p \neq w$

$\mathcal{R}_p^{(h)}$	Subset of states of submodel h mapped to reachable state space partition p for $h = 1, \ldots, H$, $p = 0, \ldots, N-1$, $\mathcal{R}_p = \times_{h=1}^{H} \mathcal{R}_p^{(h)}$, $\mathcal{R}_p^{(h)} \subseteq \mathcal{S}^{(h)}$
$\mathcal{S}$	Product state space, $\mathcal{S} = \times_{h=1}^{H} \mathcal{S}^{(h)}$
$\mathcal{S}^{(h)}$	State space of submodel h for $h = 1, \ldots, H$

Vectors

$\mathbf{0}$	Row vector of 0s
$\mathbf{e}$	Column vector of 1s
$\mathbf{e}_i$	ith column of I
$\mathbf{e}_{(it)}$	Error vector at iteration it
$\mathbf{i}$	H-dimensional state vector $(i_1, \ldots, i_H)$
$\boldsymbol{\pi}$	Steady-state probability distribution vector of Q
$\boldsymbol{\pi}_t$	Probability distribution vector of Q at time t
$\boldsymbol{\pi}_0$	Initial probability distribution vector of Q at time 0
$\boldsymbol{\pi}_{(it)}$	Probability distribution vector of Q at iteration it
$\boldsymbol{\pi}(\mathcal{R}_p)$	Probability distribution subvector of Q associated with $\mathcal{R}_p$
$\mathbf{r}_{(it)}$	Residual vector at iteration it
$\mathbf{v}^{(k)}$	State change vector of transition k for $k = 1, \ldots, K$

Matrices

I_m	Identity matrix of order m
P	One-step transition probability matrix associated with $\mathcal{R}$
Q	Infinitesimal generator matrix associated with $\mathcal{R}$
Q_D	Diagonal matrix corresponding to the diagonal of Q
Q_O	Off-diagonal part of Q, $Q_O = Q - Q_D$
$Q(p, w)$	Block (p, w) of Q for $p, w = 0, \ldots, N-1$, $Q(\mathcal{R}_p, \mathcal{R}_w)$
$Q_k^{(h)}$	Transition matrix of submodel h associated with transition k for $h = 1, \ldots, H$, $k = 1, \ldots, K$
$Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$	Transition submatrix of submodel h associated with transition k for $h = 1, \ldots, H$, $k = 1, \ldots, K$ mapped to block $Q(p, w)$ for $p, w = 0, \ldots, N-1$

Chapter 1
Introduction

This book is about the numerical modeling and analysis of systems that have very large discrete state spaces, including those with countably infinite ones, satisfying Markovian assumptions. Such systems are often composed of multiple subsystems that interact through various transitions in a well-defined manner. In this setting, the modeling of the system at hand is typically based on representing the *state* of each subsystem with what is called a *state variable* and collecting the state variables in a *state vector*. Thereafter, to each subsystem corresponds a *submodel* that is associated with a particular dimension of the state vector. Since there are as many dimensions of the state vector as there are submodels, such models are referred to as being *multidimensional*. The state space of the multidimensional model is then given by some variation of the *Cartesian product* of the state variables. Note that state variables, that is, submodels, themselves may also exhibit complicated behavior necessitating their modeling using multiple smaller submodels, which implies even a larger number of submodels, hence, dimensions in the model. We assume that one has a model of a system with submodels identified at a sufficient level of detail as such to facilitate numerical analysis as efficiently and accurately as possible.

We expand on [101] by reviewing the progress made regarding the numerical modeling and analysis of multidimensional Markovian systems represented compactly using Kronecker products of the smaller submodel transition matrices. Here, the particular emphasis is on multidimensional models having unreachable states that appear in the product state space of submodels but can never be occupied in practice. Today, the Kronecker-based approach not only enables the compact storage of the underlying state transition matrix *exactly*, but as recently shown it can also be used to reduce the memory allocated to solution vectors associated with the model during analysis. In this way, it becomes possible to numerically model and analyze

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_1

multidimensional systems that are much larger than those that can be handled with conventional sparse matrix techniques on the same platform due to memory limitations.

One particular area in which Kronecker products are being extensively used today is machine learning. As pointed out in [216], Kronecker products are very useful, for instance, in modeling networks. We should emphasize that the problem therein, and the machine learning area in general, is of an inverse nature, trying to fit a mathematical model involving Kronecker products to collected data associated with networks as accurately and as compactly as possible. On the other hand, the problem we are interested in is to model discrete state systems composed of interacting subsystems in a well-defined manner through various transitions *exactly* using Kronecker products. This modeling problem is of a forward nature and does not involve the kind of approximation that arises in the inverse problem tackled in machine learning. In our case, the only approximation in the modeling stage on computer takes place when we truncate a countably infinite state space to obtain a finite one.

The subject matter is interdisciplinary and at the intersection of applied mathematics, specifically numerical linear algebra and computational probability and computer science. Its understanding requires basic linear algebra, probability, and discrete mathematics, plus high-level programming knowledge in order to be able to follow the software implementation of methods discussed in the book. The exposition is relatively concise and rigorous, yet it tries to be complete and touches many aspects without being too technical. Since Kronecker notation necessitates the introduction of many indices on variables and is not easy to follow, the reader is walked through various examples in the text to make the discussion more accessible and concrete. The examples come from areas as diverse as systems availability, queueing networks, production lines, communication protocols, and systems of stochastic chemical kinetics. Some examples also require the employment of concepts from stability theory to aid in truncating the respective countably infinite state spaces for steady-state analysis purposes. Markov population models fall under the last class of examples and can be analyzed similarly. Up until recently, stochastic simulation seemed to be the only viable approach that would yield relatively accurate results for this class of problems. With the existing continuous improvement in computer technology, it only makes sense to invest in state-based modeling and numerical analysis approaches for very large Markovian models as those discussed here to obtain more accurate results at a finer level of detail.

The book mentions a sufficient number of modeling formalisms using Kronecker products, comparing and contrasting them along the way, so that the reader should have an idea as to how to proceed after having finished reading it. The code used in modeling and analyzing problems with Kronecker products is made publicly available. When space is not sufficient to discuss a particular issue in detail, references to the literature are provided for further reading. Open research areas are also indicated throughout the text as they become pertinent. Having set the stage with our motivation to gather all this information in one place and write the book, now we can start our discussion.

We consider discrete state space Markovian processes called *Markov chains* (MCs). These are stochastic processes which start in a particular state, evolve by visiting states in their state spaces using the available transitions, and exhibit the *memoryless* property. This property dictates that the probability distribution of the next state of the process conditioned on the current state and the previous states the process has visited depend only on the current state. In *discrete-time*, this requires that the time spent in a state during a visit be *geometrically* distributed, whereas in *continuous-time*, it requires that the time spent in a state during a visit be *exponentially* distributed.

Throughout this book, we mainly consider *continuous-time MCs* (CTMCs) and briefly touch *discrete-time MCs* (DTMCs) wherever appropriate. The list of symbols used in the book is given at the beginning. We use calligraphic letters for sets, capital letters for matrices, and boldface lowercase letters for vectors. We adhere to the convention that probability vectors are row vectors; otherwise, all vectors are column vectors as in linear algebra. The lengths of the vectors are determined by the context in which they are used. $\mathbf{e}$ represents a column vector of 1s, and $\mathbf{e}_i$ represents the ith column of the identity matrix, I. The Greek letter $\boldsymbol{\pi}$ is used to denote a particular vector. $\mathbb{1}$ denotes the indicator function which evaluates to 1 if its argument is true, else it evaluates to 0. $O(\cdot)$ stands for the big O notation. Otherwise, we use Greek letters to denote problem-specific real-valued parameters. We indicate entries of matrices using lowercase letters and the row/column indices in parentheses. Similarly, an entry of a vector is shown by writing the corresponding index in parentheses. We start indices from 0, by which it is possible to represent an empty (sub)model. An exception is state vectors, whose entries are state variables indicated by subscripts starting from 1. diag($\mathbf{a}$), subdiag($\mathbf{a}$), and supdiag($\mathbf{a}$) denote matrices with the entries of vector $\mathbf{a}$ along their diagonal, subdiagonal, and superdiagonal, respectively; all other entries of these matrices are zero. We use a subscript under I to indicate its order. Similarly, the subscript $m \times n$ under a matrix indicates that the matrix is $(m \times n)$. $\mathbf{a}(i)$ is the value of vector $\mathbf{a}$ at state i, and $\mathbf{a}(\mathcal{I})$ is the subvector of $\mathbf{a}$ associated with the states in $\mathcal{I}$. $A(\mathcal{I}, \mathcal{J})$ is the submatrix of A associated with row states in $\mathcal{I}$ and column states in $\mathcal{J}$; nz_A denotes the number of nonzeros in A. The sets of reals, nonnegative reals, integers, nonnegative integers, and positive integers are denoted by $\mathbb{R}$, $\mathbb{R}_{\geqslant 0}$, $\mathbb{Z}$, $\mathbb{Z}_{\geqslant 0}$, and $\mathbb{Z}_{>0}$, respectively.

For the time being, we assume that the state space of the CTMC is finite but later relax this assumption. So, now let the cardinality of the state space be n. A CTMC with n states may be represented by an $(n \times n)$ matrix $Q \in \mathbb{R}^{n \times n}$. This matrix is known as the *infinitesimal generator* (or *transition rate*) matrix of the associated Markovian process [165, 305]. The off-diagonal entries of this matrix are nonnegative and indicate the rates of exponentially distributed transition times between pairs of states. In other words, $q(i, j)$ for $i \neq j$ denotes the rate of exponentially distributed time by which the process makes a transition from state i to state j. The diagonal of Q is formed by the negated row sums of its off-diagonal entries (i.e., $q(i, i) = -\sum_{j \neq i} q(i, j)$) and,

hence, is nonpositive. Throughout our discussion, we assume that entries of Q are independent of time; that is, Q is *time-homogeneous*. Treatment of time-inhomogeneous CTMCs is beyond the scope of this book.

Example 1. Here is a generator matrix associated with a CTMC of 18 states

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{array} \\ \left(\begin{array}{cccccccccccccccccc} * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & & \\ \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & \\ & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & & & & & & & \\ \mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & & & & & & & \\ & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & & & & & & & \\ & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 & & & & & & \\ \mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & \\ & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & \\ & & \mu_1 & & & & & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & \\ & & & \mu_1 & & & \mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & \\ & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & \\ & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 \\ & & & & & & \mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & \\ & & & & & & & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & \\ & & & & & & & & \mu_1 & & & & & \mu_3 & * & & & \lambda_2 \\ & & & & & & & & & \mu_1 & & & \mu_2 & & & * & \lambda_3 & \\ & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 \\ \mu & & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * \end{array} \right) \end{array}, \quad (1.1)$$

where $*$ denotes the negated off-diagonal row sums. Note that Q has 67 nonzero off-diagonal entries among 306 possible ones; hence, it would normally be classified as a *sparse* matrix [93, 130, 277, 280, 305, 310].

Sparse matrices can be stored in many different formats, but the two most commonly used ones for our purposes are the coefficient sparse format and the row sparse format. In the *coefficient sparse format*, nonzeros of the matrix, their row indices, and their column indices are stored in the same order in a floating-point array `a`, an integer array `ia`, and an integer array `ja`, each of length equal to the number of nonzeros in Q. In the *row sparse format*, the `a` and `ja` arrays are as in the coefficient sparse format with the constraint that nonzeros must have been stored row by row starting from the lowest indexed row up to the highest indexed row. The array `ia` is of length $n + 1$ and stores the starting indices of rows in `a` and `ja`. For instance, `ia(i)` gives the starting index of row `i` in `a` and `ja` so that `ia(i+1) - ia(i)` is the number of nonzeros in row `i`.

It is well known that the minimum of exponentially distributed random variables is exponentially distributed with rate parameter equal to the sum of the rate parameters of the random variables. Therefore, the absolute value of a

diagonal entry may be perceived as the rate of the exponentially distributed exit time from that particular state. This implies that row sums of Q are necessarily 0.

Let the *initial* probability distribution vector of Q be denoted by $\boldsymbol{\pi}_0$. Clearly, $\boldsymbol{\pi}_0 \in \mathbb{R}^{1\times n}_{\geqslant 0}$ and $\boldsymbol{\pi}_0\mathbf{e} = 1$. The ith entry of $\boldsymbol{\pi}_0$, written $\boldsymbol{\pi}_0(i)$, denotes the probability of the process being in state i at time 0. Then the *transient* probability distribution vector $\boldsymbol{\pi}_t \in \mathbb{R}^{1\times n}_{\geqslant 0}$ at time $t \in \mathbb{R}_{\geqslant 0}$ satisfies

$$\frac{d\boldsymbol{\pi}_t}{dt} = \boldsymbol{\pi}_t Q, \quad \boldsymbol{\pi}_t\mathbf{e} = 1 \tag{1.2}$$

and is given by

$$\boldsymbol{\pi}_t := \boldsymbol{\pi}_0 e^{Qt}.$$

Transient solution methods are based on computing the *matrix exponential* e^{Qt} explicitly or implicitly [305]. Since computing e^{Qt} explicitly is not feasible when Q is large and sparse, we will concentrate on methods that are able to follow the latter approach. In passing to steady-state analysis, we remark that almost all methods for transient analysis of large and sparse MCs are based on vector–matrix multiplications.

There are processes for which $\lim_{t\to\infty} \boldsymbol{\pi}_t$ exists. Such processes are said to be *ergodic*. Whenever this limit exists and therefore the process is ergodic, we define $\boldsymbol{\pi} := \lim_{t\to\infty} \boldsymbol{\pi}_t$ and refer to $\boldsymbol{\pi} \in \mathbb{R}^{1\times n}_{\geqslant 0}$ as the *steady-state* (or *limiting*, *long-run*) probability distribution vector. The steady-state vector satisfies

$$\boldsymbol{\pi}Q = \mathbf{0}\,, \quad \boldsymbol{\pi}\mathbf{e} = 1\,, \tag{1.3}$$

and, hence, is also a *stationary* distribution [165, 305]. For finite CTMCs, the necessary and sufficient condition for ergodicity is *irreducibility*, that is, each state must be reachable from every other state by following the nonzero transitions in Q. Many properties of transition matrices underlying finite MCs can be found in [30, 194, 290]. Conditions for ergodicity in irreducible CTMCs with countably infinite state spaces are given later in the book.

In the Kronecker-based approach, Q is represented exactly using Kronecker products [92, 260, 320] of smaller matrices and is never explicitly generated. This is motivated by the fact that models of systems are usually comprised of interacting submodels and, hence, have multidimensional state spaces. Each submodel can be considered as representing a separate dimension in this multidimensional state space. The state of a submodel may either evolve independently, that is, in isolation from other submodels, or it may evolve in synchronization with other submodels. To the former case, there corresponds a Kronecker product with all factors except the one associated with the particular submodel being identity matrices. To the latter case, there corresponds a Kronecker product with all factors, except the ones associated with the submodels that need to be synchronized, being identity matrices. Hence, Q can be represented using sums of such Kronecker products. The im-

plementation of transient solvers for (1.2) and steady-state solvers for (1.3) can rest on this compact Kronecker representation, thanks to the existence of an efficient vector–Kronecker product multiplication algorithm known as the *shuffle algorithm* [263].

Example 1. (ctnd.) The CTMC corresponding to (1.1) is the model of a system [15] with three submodels as in Figure 1.1 having, respectively, two, one, and two components, where only one of which is working in each submodel.

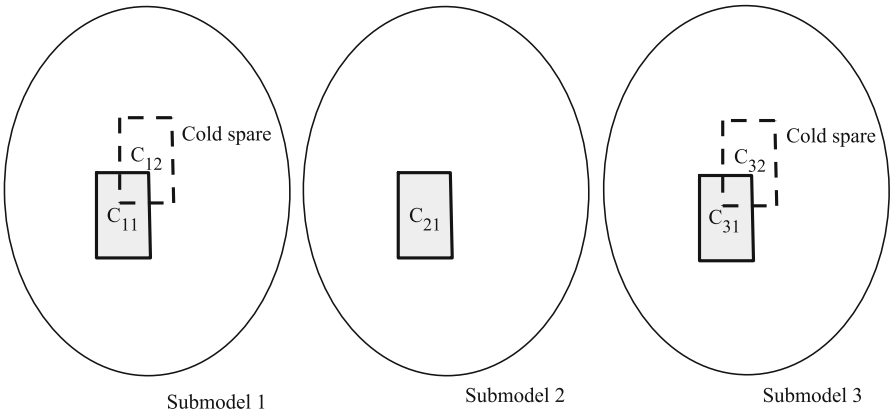


Fig. 1.1 Simple availability model with three submodels.

The non-working component(s) in each submodel act as cold spare(s). The working component in each submodel fails independently of other submodels with rates of λ_1, λ_2, and λ_3, respectively. When a working component fails, it is replaced in no time with a cold spare (if there is any left) in the same submodel. Furthermore, there is one repairman in each submodel who can repair a failed component independently of other submodels with rates of μ_1, μ_2, and μ_3, respectively. When all components fail, the model is brought up by a global repair with rate μ.

To give some examples, in (1.1) state 0 corresponds to all submodels being intact, state 17 corresponds to all submodels having failed, and state 14 corresponds to failed first and third submodels, while the second submodel is intact. Overall, this is a model which tries to improve steady-state availability by using a larger number of redundant components, which themselves do not need to be highly reliable.

In practice, the representation of Q based on Kronecker products is obtained using various modeling formalisms. These include compositional Markovian models such as stochastic automata networks (SANs) [263, 264, 265, 305] and different classes of superposed stochastic Petri nets [129, 195], hierarchical Markovian models (HMMs) of queueing networks [43], generalized stochastic Petri nets (GSPNs) [68], or systems of asynchronously com-

municating stochastic modules [73], and stochastic process algebras such as the performance evaluation process algebra (PEPA) [186]. These modeling formalisms are integrated to various software packages such as the Abstract Petri Net Notation (APNN) toolbox [7, 22], the Performance Evaluation of Parallel Systems (PEPS) software tool [25, 38, 138, 259] (see also GTAexpress [89]), the PEPA Workbench [83, 258], and the Stochastic Model checking Analyzer for Reliability and Timing (SMART) [81].

An advantage of HMMs is their ability to represent Q using Kronecker products without introducing unreachable states. Matrix diagrams [82] and representations for specific models as in [179] can also be used to achieve the same effect when state spaces are expressed compositionally. There are other approaches that can be used to deal with unreachable states as discussed in [26, 49, 57]. At the beginning of our discussion, we make the assumption that the product state space of the model composed of submodels does not have unreachable states and is irreducible. Later we relax this assumption and show how this problem can be tackled using the HMM approach. Yet, in many practical applications, Q is very large and has many nonzeros necessitating its storage in memory using Kronecker products. In order to analyze Markovian models based on Kronecker products efficiently, various algorithms for vector–Kronecker product multiplication based on the shuffle algorithm are devised [26, 27, 57, 88, 110, 136, 137, 265, 266] and used as kernels in iterative solution techniques proposed for different modeling formalisms. The transient distribution in (1.2) can be computed using vector–Kronecker product multiplications as in [43]. The steady-state distribution in (1.3) also needs to be computed using vector–Kronecker product multiplications, since *direct methods* based on complete factorizations, such as LU factorization through Gaussian elimination (GE) [125, 158, 302, 305], normally introduce new nonzeros which cannot be accommodated [100].

The two papers [50, 307] provide good overviews of iterative solution techniques for the analysis of MCs based on Kronecker products. Solution of lower- and upper-triangular systems of linear equations arising from a complete factorization of a Kronecker product is the subject of [135]. The approach relies on factorizing the smaller factors making up the Kronecker product. The problem with this approach is that even if the factorization existed, it is not clear why the resulting smaller lower- and upper-triangular factors should also be sparse. Issues related to reachability analysis, vector–Kronecker product multiplication, hierarchical state space generation in Kronecker-based matrix representations for large Markovian models are surveyed in [70]. A comparison of the merits of the SAN and GSPN modeling formalisms using the PEPS and SMART software packages can be found in [80]. Along a different line, [91, 142, 146] give conditions for SANs to have product–form steady-state distributions, whereas transient availability of a grid of 5,000 CPUs is investigated in [37] using SANs. Finally, graph-theoretic arguments are used in [147] to aid the steady-state analysis of SANs.

Although Kronecker representations for CTMCs underlying many models of practical applications have been considered, so far only a handful of DTMCs based on Kronecker products appeared in the literature. For instance, the one in [267] is a model of synchronization via message passing in a distributed system, the one in [264] is a model of the mutual exclusion algorithm of Lamport, those in [144, 145] are models of buffer admission mechanisms for asynchronous transfer mode (ATM) networks from telecommunications, the one in [323] is a multiservices resource allocation policy for wireless ATM networks, and the one in [141] is a model to compute the loss rate in a multistage ATM switch. The model in [175] is a larger, scalable, and extended version of that in [323]. It serves as a good example showing that the underlying MC of a discrete-time model based on Kronecker products can be relatively dense and numerically difficult to analyze. These case studies are based on the SAN modeling formalism, whereas [286] extends the Kronecker representation to stochastic Petri nets with discrete phase-type distributions. Sufficient conditions for DTMCs with transition probability matrices in the form of generalized Kronecker products to have product–form solutions are investigated in [143]. Clearly, the area of DTMCs based on Kronecker products can use other case studies and formalisms.

Having been able to represent the generator matrix compactly using Kronecker products for many years, the challenge in Kronecker-based Markovian analysis was to perform vector–Kronecker product multiplication in as fast and as little of memory possible. This operation is known to be executed efficiently by the shuffle algorithm [136] when the factors in the Kronecker product terms are relatively dense. On the other hand, it may be more efficient to obtain nonzeros of the generator matrix in Kronecker form on the fly and multiply them with corresponding elements of the vector [57] when the factors are relatively sparse. Inspired by these observations, recently the shuffle algorithm has been improved to avoid unnecessary *floating-point operations* (flops) that evaluate to zero during the course of the multiplication and possibly reduce the amount of memory used [110]. This is done by exploiting zero rows and columns in the smaller submodel transition matrices. It has been shown that in many cases the *modified shuffle algorithm* performs a smaller number of flops than the shuffle algorithm and the algorithm that generates nonzeros on the fly, sometimes with a minimum number of flops and as little of memory possible. Nevertheless, all of these vector–Kronecker product multiplication algorithms still require the allocation of *full vectors* of length equal to the reachable state space size, which becomes the bottleneck in analysis due to the exponential increase in the sizes of the vectors with the increasing number of dimensions.

With the recent advances in storing and analyzing dense multidimensional data numerically [177], a different approach for reducing memory allocated to vectors in Kronecker-based Markovian analysis has been proposed [64]. As opposed to the full-vector approach described above, we refer to this as the *compact vector* approach since Kronecker products of shorter vectors

are used to represent data with a user controllable accuracy [163, 178, 202, 203, 256]. In other words, essential information that exists in the data of a full vector is extracted and stored using a number of shorter vectors by the help of a particular Kronecker decomposition at the expense of some approximation error that can be bounded. This approach has paved the way for an altogether different line of research [63, 64, 66, 117, 118] that enables us to investigate the compactness of the proposed Kronecker representation for vectors in comparison to the quality of the solution obtained during analysis.

Here we take a vector–matrix approach in discussing recent results related to the modeling and analysis of multidimensional MCs based on Kronecker products. The book may be viewed as consisting of two parts. The first part is formed of Chapter 2 through Chapter 4 which provide the necessary background to follow the developments in the field regarding the Kronecker-based modeling of multidimensional MCs. The second part concentrates on numerical analysis methods for such representations and is formed of Chapter 5 through Chapter 7. The analysis part is based on concepts introduced in the modeling part but can still be mostly read by itself. In fact, this was also the premise of [101] which is primarily on analysis.

The first part of the book starts with Chapter 2, which provides background information on Kronecker operations and explains in detail the Kronecker-based SAN and HMM modeling formalisms used in representing the generator matrix compactly on the small CTMC in Chapter 1, another small example but having unreachable states, and two larger example models from the areas of systems availability and queueing networks. We adopt a Kronecker-based representation similar to that of HMMs which concentrates on the reachable state space and uses sums of Kronecker products but intend to keep the discussion independent from modeling formalisms so that it is easier to follow and tailor. In the same chapter, we show that the generator matrix underlying a multidimensional CTMC has a rich structure which is nested and recursive. Having covered SANs and HMMs and argued why representing the reachable state space as in HMMs is important, Chapter 3 looks into the problem of representing the reachable state space of a multidimensional MC as a union of Cartesian products of subsets of submodel state spaces. In doing this, it considers a merge-based approach and a refinement-based approach. In this chapter, two other example models from the areas of production lines and communications protocols are introduced. The second example which is modeled using a GSPN is investigated in detail from a Kronecker-based perspective. Again results of experiments on problems that compare and contrast these approaches are reported. Chapter 4 considers preprocessing of the Kronecker representation so as to expedite numerical analysis. We discuss permuting the nonzero structure of the generator matrix underlying the CTMC symmetrically by reordering, changing the orders of the nested blocks by grouping, and reducing the size of the state space by lumping. The effects of reordering [100] are demonstrated through the LU factorization [158, 302, 305] of diagonal blocks in the nested partitioning as-

sociated with the Kronecker structure on the communications protocol model introduced earlier. This approach is related to taking advantage of common Schur factors in diagonal blocks. It is in this chapter that we also discuss how the countably infinite state space of a multidimensional CTMC can be truncated judiciously using Lyapunov functions for steady-state analysis. Three example models from the areas of systems of stochastic chemical kinetics and queueing networks are used to show how this procedure can be performed in practice. The third example in Chapter 4 serves as a relatively complicated case study.

In the second part of the book, Chapter 5 is about the shuffle and modified shuffle algorithms used in the full-vector approach for vector–Kronecker product multiplication. The two algorithms are compared on the small example, and results of numerical experiments on larger problems are reported from the literature. This chapter also discusses the recent developments in representing solution vectors compactly and how vector–Kronecker product multiplication can be performed in that context. Chapter 6 is about the steady-state analysis of multidimensional CTMCs based on Kronecker products with block iterative, preconditioned projection, multilevel, decompositional, and matrix analytic methods. For all the iterative methods discussed, it is essential to consider splitting the smaller matrices corresponding to submodels. The examples introduced in Chapters 2, 3, and 4 are again used to illustrate various concepts throughout the discussion on iterative methods. Chapter 6 closes by touching issues that need to be addressed when compact solution vectors are used in the implementation of iterative methods. Chapter 7 is devoted to the transient analysis of multidimensional CTMCs based on Kronecker products using uniformization, explicit and implicit ordinary differential equation (ODE) solvers. When the model has a countably infinite state space, it must be truncated at each time step until the horizon is reached by dynamically expanding the set of important states using reachability information and probabilistic arguments, solving for the transient distribution of the particular set of states at that time step and then shrinking the set so that it once again has only important states to be considered at the next time step. We should remark that although explicit transient solvers can be implemented using vector–Kronecker product multiplication alone, implicit transient solvers need to use iterative linear system solvers similar to those in Chapter 6. A discussion on how the chemical master equation can be tackled using an implicit ODE solver with compact solution vectors appears at the end of Chapter 7. Chapter 8 provides the conclusion.

Our intention is for the book to be read sequentially. The reachable state space representation in Chapter 3 happens to be at the heart of the discussion. All techniques that follow in Chapter 4 through Chapter 7 are based on understanding how the reachable state space is represented on computer. Knowing the SAN formalism in Chapter 2 alone will not help to understand this essential part of the book, because SANs do not differentiate between reachable and unreachable states and end up representing both. In this sense,

one cannot just skip some of the material, such as the section on HMMs in Chapter 2, and delve into a particular chapter randomly. In order to benefit from the discussion, one should definitely read and understand Chapter 3 after Chapter 2. Furthermore, one who does not follow Chapter 3 closely will have difficulty implementing any of the methods discussed after Chapter 3. Similarly, understanding the analysis methods in Chapter 6 and Chapter 7 requires one to understand Chapter 5 and the Kronecker product operation introduced at the beginning of Chapter 2 thoroughly. Along this line, we should add that the discussion in this book is also more advanced compared to that in [101] since the techniques in this book are presented for models whose reachable state spaces need not be equal to their product state spaces. Although [101] could be understood by knowing SANs alone, this book cannot be.

All numerical experiments in the book are performed on an Intel Core i7 2.6 GHz processor with 16 GB main memory under Linux. The results can be extended to DTMCs based on Kronecker products with some modifications. Areas that need further research are mentioned within the sections as they are discussed. Finally, we remark that the subject of parallel implementations exploiting the Kronecker representation of MCs is briefly touched in Chapters 6 and 7, but it is still an open area for research deserving specialized treatment.

Chapter 2
Modeling with Kronecker Products

The *Kronecker* (or *tensor*) *product* of two (rectangular) matrices A and B with $A = [a(i_A, j_A)]$ is

$$A \otimes B = [a(i_A, j_A)B] \, .$$

Or more formally, given $A \in \mathbb{R}^{n_A \times m_A}$ and $B \in \mathbb{R}^{n_B \times m_B}$, $A \otimes B$ yields the (rectangular) matrix $C \in \mathbb{R}^{n_A n_B \times m_A m_B}$ whose entries satisfy

$$c(i_C, j_C) = a(i_A, j_A) b(i_B, j_B)$$

$$\text{with} \quad i_C = i_A n_B + i_B \quad \text{and} \quad j_C = j_A m_B + j_B$$

for

$$(i_A, j_A) \in \{0, \ldots, n_A - 1\} \times \{0, \ldots, m_A - 1\} \, ,$$
$$(i_B, j_B) \in \{0, \ldots, n_B - 1\} \times \{0, \ldots, m_B - 1\} \, .$$

Here, $\otimes$ is the *Kronecker product* [92, 320] operator and $\times$ is the *Cartesian product* [162] operator. Note that in a two-dimensional representation, the row indices of C are in $\{0, \ldots, n_A - 1\} \times \{0, \ldots, n_B - 1\}$, whereas its column indices are in $\{0, \ldots, m_A - 1\} \times \{0, \ldots, m_B - 1\}$. Hence, the ordering of rows and columns of C with respect to this two-dimensional representation is *lexicographical*, since

$$c(i_C, j_C) = c((i_A, i_B), (j_A, j_B)) = c(i_A n_B + i_B, j_A m_B + j_B) \, .$$

Kronecker product is *associative* and defined for more than two matrices. To explain this further for a MC setting, let us consider the Kronecker product of H square matrices as in

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_2

$$X = X^{(1)} \otimes \cdots \otimes X^{(H)} = \bigotimes_{h=1}^{H} X^{(h)} \ ,$$

where $X^{(h)} \in \mathbb{R}^{n_h \times n_h}$ and row/column indices of $X^{(h)}$ are in $\mathcal{S}^{(h)} = \{0, \ldots, n_h - 1\}$ for $h = 1, \ldots, H$. Therefore, $X \in \mathbb{R}^{n \times n}$ with $n = \prod_{h=1}^{H} n_h$, and the ordered H-dimensional tuples

$$\mathbf{i} = (i_1, \ldots, i_H) \in \bigtimes_{h=1}^{H} \mathcal{S}^{(h)} \quad \text{and} \quad \mathbf{j} = (j_1, \ldots, j_H) \in \bigtimes_{h=1}^{H} \mathcal{S}^{(h)}$$

may be used to represent the row and column indices of X, respectively. Hence, Kronecker product of H square matrices implies a one-to-one onto mapping between an H-dimensional state space and a one-dimensional (or *flat*) state space that are lexicographically ordered, and naturally, Kronecker products have been used to define the transition matrices of MCs with multidimensional state spaces.

Having defined the Kronecker product, we are in a position to introduce the *Kronecker* (or *tensor*) *sum* of two square matrices. Given $A \in \mathbb{R}^{n_A \times n_A}$ and $B \in \mathbb{R}^{n_B \times n_B}$, their Kronecker sum $A \oplus B$ yields the square matrix $C \in \mathbb{R}^{n_A n_B \times n_A n_B}$ which satisfies

$$C = A \oplus B = A \otimes I_{n_B} + I_{n_A} \otimes B \ .$$

Here, $\oplus$ is the *Kronecker* sum operator. More generally, Kronecker sum of H square matrices $X^{(h)} \in \mathbb{R}^{n_h \times n_h}$ as in

$$X^{(1)} \oplus \cdots \oplus X^{(H)} = \bigoplus_{h=1}^{H} X^{(h)}$$

satisfies

$$\bigoplus_{h=1}^{H} X^{(h)} = \sum_{h=1}^{H} I_{\prod_{l=1}^{h-1} n_l} \otimes X^{(h)} \otimes I_{\prod_{l=h+1}^{H} n_l} \ .$$

As it can be seen from the last identity, in fact Kronecker sum is a summation whose H terms are formed for $h = 1, \ldots, H$ using two Kronecker product operators separating the identity matrix of order $\prod_{l=1}^{h-1} n_l$, the square matrix $X^{(h)}$, and the identity matrix of order $\prod_{l=h+1}^{H} n_l$, respectively. As such, the first term lacks the identity matrix to the left of $X^{(1)}$, and the last term lacks the identity matrix to the right of $X^{(H)}$. This can be easily validated from the identity used in the definition of the Kronecker sum of two matrices A and B. Although it is not needed as a computational tool in our analyses, Kronecker sum enables the composition of local transitions of submodels in our modeling framework as we will see in the next section.

In passing to our discussion on modeling with Kronecker products, we remark that an important property of a Kronecker product is its *compatibility* with matrix multiplication under certain conditions. That is, if A, B, C, and D are matrices of suitable sizes so that AC and BD exist, then

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \ .$$

We will see the usefulness of this property in the design of the shuffle algorithm in Chapter 5.

2.1 Stochastic Automata Networks

We consider a model of a system with submodels identified at a sufficient level of detail to facilitate analysis as discussed at the beginning of Chapter 1. In a *stochastic automata network* (SAN) [263, 264, 265, 305], each submodel of the Markovian model at hand is represented by a *stochastic automaton*. Before we introduce the definition of the generator matrix of a SAN using Kronecker sums and Kronecker products, let us explain the two kinds of transitions that affect the automata to motivate these operations.

When automata associated with submodels are non-interacting, description of each automaton consists of local transitions only. Each *local transition* changes the state of the particular automaton with which it is associated and has no influence on the state of other automata. Interactions among submodels are captured by *synchronizing transitions*. Synchronization among automata happens when a state change in one automaton causes a simultaneous state change in other automata. As we will see, local transitions and synchronizing transitions in SANs yield Kronecker sums and Kronecker products, respectively.

A continuous-time Markovian model of H submodels can be represented by associating stochastic automaton $\mathcal{A}^{(h)}$ defined on the state space $\mathcal{S}^{(h)} = \{0, \ldots, n_h - 1\}$ with submodel h for $h = 1, \ldots, H$. Local transitions of $\mathcal{A}^{(h)}$ are represented by *local transition rate matrix* $Q_l^{(h)} \in \mathbb{R}^{n_h \times n_h}$ with row sums of 0 for $h = 1, \ldots, H$. The diagonal entries of $Q_l^{(h)}$ are equal to their negated off-diagonal row sums. When there are K synchronizing transitions in the model, $\mathcal{A}^{(h)}$ has the *synchronizing transition matrix* $Q_{e_k}^{(h)} \in \mathbb{R}_{\geqslant 0}^{n_h \times n_h}$, which represents the contribution of $\mathcal{A}^{(h)}$ to synchronizing transition e_k and the corresponding diagonal corrector matrix $\bar{Q}_{e_k}^{(h)} \in \mathbb{R}^{n_h \times n_h}$ for $k = 1, \ldots, K$.

The automaton that triggers a synchronizing transition is called the *master*, and the others that get affected by the synchronizing transition are called the *slaves* [136]. Nevertheless, many times automata are not in a master–slave relationship from the point of view of the synchronizing transition but act together in the manner of a rendezvous to realize the transition. Yet, in the SAN modeling formalism, one of the automata that participates in a syn-

chronizing transition is identified as the master and the other(s) that are affected as the slave(s) in order to specify the matrices associated with the transition. With this understanding, synchronizing transition matrices are either transition rate matrices corresponding to master automata or transition probability matrices corresponding to slave automata. If $\mathcal{A}^{(h)}$ for some h is not involved in synchronizing transition e_k, then $Q_{e_k}^{(h)} = \bar{Q}_{e_k}^{(h)} = I_{n_h}$, where $n_h = |\mathcal{S}^{(h)}|$ and I_{n_h} is the identity matrix of order n_h.

The generator matrix corresponding to the SAN model is then given by

$$Q = Q_l + Q_e + \bar{Q}_e, \tag{2.1}$$

where

$$Q_l = \bigoplus_{h=1}^{H} Q_l^{(h)}, \quad Q_e = \sum_{k=1}^{K} \bigotimes_{h=1}^{H} Q_{e_k}^{(h)}, \quad \bar{Q}_e = \sum_{k=1}^{K} \bigotimes_{H=1}^{H} \bar{Q}_{e_k}^{(h)}.$$

The Kronecker representation in (2.1) associated with the generator matrix is referred to as the *descriptor* of the SAN. Here, the third term of the descriptor, $\bar{Q}_e$, ensures that the row sums of Q are zero.

The H-dimensional state space of the CTMC underlying Q is given by

$$\mathcal{S} = \bigtimes_{h=1}^{H} \mathcal{S}^{(h)} ,$$

the *product state space*. Observe that $|\mathcal{S}| = \prod_{h=1}^{H} |\mathcal{S}^{(h)}| = \prod_{h=1}^{H} n_h = n$. Furthermore, the one-dimensional value of state $\mathbf{i} = (i_1, \ldots, i_H) \in \mathcal{S}$, where $i_h \in \mathcal{S}^{(h)}$ for $h = 1, \ldots, H$, may be obtained from

$$i = \sum_{h=1}^{H} i_h \prod_{l=h+1}^{H} n_l \, .$$

When implementing an algorithm, many times one needs to have a mapping from the one-dimensional state space to the multidimensional state space, and vice versa, since solution vectors are one-dimensional and appropriate entries of them need to be accessed. Therefore, we will be using the one-dimensional and multidimensional representations of states interchangeably throughout the book.

In general, there is no canonical form for SANs, and one has flexibility in designating the master and the slave(s) of a synchronizing transition and assigning real numbers to the respective entries of their transition matrices so that when the real numbers are multiplied their product gives the rate corresponding to the particular transition. Without loss of generality, we assume that synchronizing transition probability matrices of a SAN have row sums of 1 or 0. In fact, each SAN can be transformed to this form which we call *proper* [173]. Observe that in this form, a row sum of 0 (that is, a zero row) in row i_h of synchronizing transition probability matrix $Q_{e_k}^{(h)}$ models the inhibition of synchronizing transition e_k when $\mathcal{A}^{(h)}$ is in state i_h.

Local transitions can be constant, meaning their rates do not depend on the state of other automata, or they can be functional. When a local transition is *functional*, its transition rate is a nonnegative real-valued function that depends on the state of other automata [136]. We remark that automata having local transitions that are functional cease to be non-interacting. Similar to local transitions, synchronizing transitions can be constant or functional. Note that if a SAN consists of only nonfunctional local transitions and does not have any synchronizing transitions, then its automata are non-interacting, and therefore, each automaton can be analyzed separately, implying a product–form probability distribution. Such models are not interesting and therefore will not be considered. On the other hand, to the contrary of what one may think, it is possible to have a SAN with no local transitions. In that case, there must be at least one synchronizing transition so that there is interaction among automata and the model is meaningful. When there are functional entries in the submodel transition matrices, Kronecker products/sums become *generalized Kronecker products/sums* [265]. We will return to functional transitions in SANs in our second example.

Example 1. (ctnd.) Let us revisit the simple availability model in Figure 1.1. This model can be specified as a SAN having three automata and one synchronizing transition (i.e., $H = 3, K = 1$) with $\mathcal{A}^{(h)}$ representing submodel h for $h = 1, 2, 3$ and state space sizes of $n_1 = 3$, $n_2 = 2$, and $n_3 = 3$.

A possible state representation for $\mathcal{A}^{(h)}$ is one in which 0 represents the state with no failed components, while $n_h - 1$ represents the state with all failed components in submodel h. Local transitions represent local component failures and repairs in each submodel, whereas the synchronizing transition represents the global repair which takes place when all submodels have failed. In this model, all transitions have constant rates associated with them as in Figure 2.1.

With this understanding, assuming $\mathcal{A}^{(1)}$ is the master for the synchronizing transition, and again letting $*$ denote the negated off-diagonal row sums, the local transition rate matrices are given by

$$Q_l^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} * & \lambda_1 & \\ \mu_1 & * & \lambda_1 \\ & \mu_1 & * \end{pmatrix} \end{array}, \quad Q_l^{(2)} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} * & \lambda_2 \\ \mu_2 & * \end{pmatrix} \end{array}, \quad Q_l^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} * & \lambda_3 & \\ \mu_3 & * & \lambda_3 \\ & \mu_3 & * \end{pmatrix} \end{array}.$$

The synchronizing transition matrices are

$$Q_{e_1}^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} & & \\ & & \\ \mu & & \end{pmatrix} \end{array}, \quad Q_{e_1}^{(2)} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} & \\ 1 & \end{pmatrix} \end{array}, \quad Q_{e_1}^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} & & \\ & & \\ 1 & & \end{pmatrix} \end{array},$$

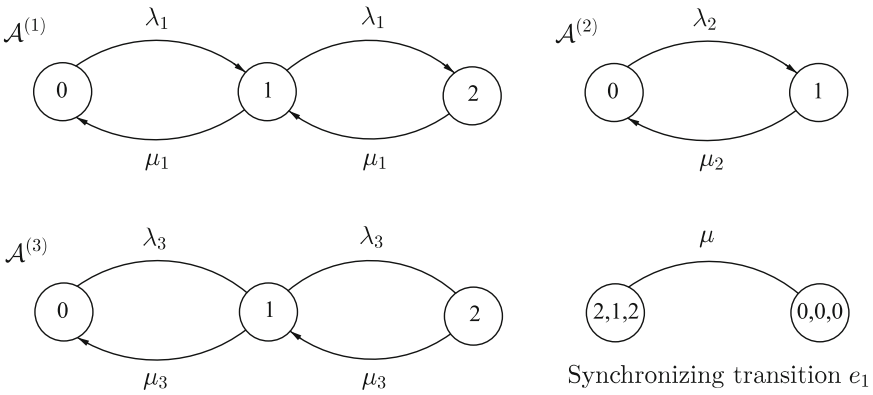


Fig. 2.1 SAN for simple availability model with three submodels

and the diagonal corrector matrices become

$$\bar{Q}_{e_1}^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} & & \\ & & \\ & & -\mu \end{pmatrix} \end{array}, \quad \bar{Q}_{e_1}^{(2)} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} & \\ & 1 \end{pmatrix} \end{array}, \quad \bar{Q}_{e_1}^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} & & \\ & & \\ & & 1 \end{pmatrix} \end{array}.$$

Several comments are due. First, note that Q_l in (2.1) can be expressed as

$$Q_l = \bigoplus_{h=1}^{3} Q_l^{(h)} = Q_l^{(1)} \otimes I_6 + I_3 \otimes Q_l^{(2)} \otimes I_3 + I_6 \otimes Q_l^{(3)}.$$

Hence, Q_l is a sum of Kronecker products in which each term has a single factor that is not identity. On the other hand,

$$Q_e = Q_{e_1}^{(1)} \otimes Q_{e_1}^{(2)} \otimes Q_{e_1}^{(3)} \quad \text{and} \quad \bar{Q}_e = \bar{Q}_{e_1}^{(1)} \otimes \bar{Q}_{e_1}^{(2)} \otimes \bar{Q}_{e_1}^{(3)} .$$

Then from (2.1), we have $Q = Q_l + Q_e + \bar{Q}_e$, a sum of five Kronecker product terms yielding (1.1).

Observe that we could have chosen a different ordering for the automata (i.e., numbered automata differently), a different state representation, (i.e., numbered states differently) in each automaton, or even the master of synchronizing transition e_1 differently. Yet, these would all lead to different symmetric permutations of the underlying generator matrix Q. This is something to which we return in Chapter 4.

Now, if we assume the absence of a matrix in a Kronecker product indicates that it is identity, then it is possible to do without storing identity matrices. Therefore, the number of floating-point values stored in the SAN representation of Q is 24, whereas it is 85 for the flat representation in (1.1).

The storage discrepancy between Kronecker and flat representations becomes substantial for larger values of the state space size, n, and for denser submodel transition matrices. We remark that it is also possible to take advantage of identity matrices in the vector–Kronecker product multiplication algorithms discussed in Chapter 5.

Now, let us assume that the SAN descriptor in (2.1) is written simply as

$$Q = \sum_{k=1}^{H+2K} \bigotimes_{h=1}^{H} Q_k^{(h)} \,. \tag{2.2}$$

Without losing generality, we assume that the first H of the $H + 2K$ Kronecker product terms represents local transitions, the kth one being that of submodel k, whereas the remaining $2K$ Kronecker product terms represents synchronizing transitions and their diagonal correctors. In each of the first H terms, there is only one matrix different than identity; for the kth term, it is the kth matrix. In each of the remaining $2K$ terms, there are at least two matrices different than the identity matrix, and those correspond to submodels that get synchronized by that particular transition. The existence of an identity matrix in a term implies that the corresponding submodel does not change its state during that particular transition. Hence, only one submodel may change its state during a local transition, and at least two submodels may change their states during a synchronizing transition. If a particular automaton does not have local transitions, then its corresponding local transition rate matrix may be set to the zero matrix, which also need not be stored.

There is a rich structure associated with the Kronecker representation in (2.2). This structure is nested and recursive [58, 59, 60, 98, 172, 174, 175, 315]. Let level 0 denote the highest level at which Q is perceived as a single block of order $n = \prod_{h=1}^{H} n_h$.

Example 1. (ctnd.) Since $n = 18$ due to $n_1 = n_3 = 3$, and $n_2 = 2$, at level 0 we have the order 18 matrix (cf. (1.1))

$$
Q = \begin{array}{c}
\begin{array}{cccccccccccccccccc}
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2
\end{array} \\
\begin{array}{c}
0\,0\,0 \\ 0\,0\,1 \\ 0\,0\,2 \\ 0\,1\,0 \\ 0\,1\,1 \\ 0\,1\,2 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,0\,2 \\ 1\,1\,0 \\ 1\,1\,1 \\ 1\,1\,2 \\ 2\,0\,0 \\ 2\,0\,1 \\ 2\,0\,2 \\ 2\,1\,0 \\ 2\,1\,1 \\ 2\,1\,2
\end{array}
\begin{pmatrix}
* & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & & \\
\mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & \\
 & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & & & & & & & \\
\mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & & & & & & & \\
 & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & & & & & & & \\
 & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 & & & & & & \\
\mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & \\
 & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & \\
 & & \mu_1 & & & & & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & \\
 & & & \mu_1 & & & \mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & \\
 & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & \\
 & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 \\
 & & & & & & \mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & \\
 & & & & & & & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & \\
 & & & & & & & & \mu_1 & & & & & * & & & \lambda_2 \\
 & & & & & & & & & \mu_1 & & & \mu_2 & & * & \lambda_3 & \\
 & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 \\
\mu & & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & *
\end{pmatrix}
\end{array}.
$$

At the next level, which we call level 1, Q is an $(n_1 \times n_1)$ block matrix with blocks of order $\prod_{h=2}^{H} n_h$.

Example 1. (ctnd.) At level 1, we have the (3×3) block matrix

$$
Q = \left(\begin{array}{cccccc|cccccc|cccccc}
* & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & & \\
\mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & \\
 & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & & & & & & & \\
\mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & & & & & & & \\
 & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & & & & & & & \\
 & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 & & & & & & \\
\hline
\mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & \\
 & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & \\
 & & \mu_1 & & & & & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & \\
 & & & \mu_1 & & & \mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & \\
 & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & \\
 & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 \\
\hline
 & & & & & & \mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & \\
 & & & & & & & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & \\
 & & & & & & & & \mu_1 & & & & & * & & & \lambda_2 \\
 & & & & & & & & & \mu_1 & & & \mu_2 & & * & \lambda_3 & \\
 & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 \\
\mu & & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & *
\end{array}\right)
$$

with blocks of order 6.

At level 2, Q is an $(n_1 n_2 \times n_1 n_2)$ block matrix with blocks of order $\prod_{h=3}^{H} n_h$.

Example 1. (ctnd.) At level 2, we have the (6×6) block matrix

$$Q = \left(\begin{array}{ccc|ccc|ccc|ccc|ccc|ccc}
* & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & & \\
\mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & \\
 & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & & & & & & & \\
\hline
\mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & & & & & & & \\
 & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & & & & & & & \\
 & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 & & & & & & \\
\hline
\mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & \\
 & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & \\
 & & \mu_1 & & & & & \mu_3 & * & & & \lambda_2 & & & \lambda_1 & & & \\
\hline
 & & & \mu_1 & & & \mu_2 & & & * & \lambda_3 & & & & & \lambda_1 & & \\
 & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 & & & & & \lambda_1 & \\
 & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & & & & & & \lambda_1 \\
\hline
 & & & & & & \mu_1 & & & & & & * & \lambda_3 & & \lambda_2 & & \\
 & & & & & & & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & \\
 & & & & & & & & \mu_1 & & & & & \mu_3 & * & & & \lambda_2 \\
\hline
 & & & & & & & & & \mu_1 & & & \mu_2 & & & * & \lambda_3 & \\
 & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & \lambda_3 \\
\mu & & & & & & & & & & & \mu_1 & & & \mu_2 & & \mu_3 & *
\end{array}\right)$$

with blocks of order 3.

Continuing in this manner, at level H, Q is a $(\prod_{h=1}^{H} n_h \times \prod_{h=1}^{H} n_h)$, in other words, $(n \times n)$, block matrix with blocks of order 1.

Example 1. (ctnd.) Finally, at level 3, we have the (18×18) block matrix with blocks of order 1 as in (1.1).

More formally, for $l = 0, \ldots, H$ we have

$$b_l = \begin{cases} 1 & \text{if } l = 0 \\ n_l^2 b_{l-1} & \text{otherwise} \end{cases} \quad \text{and} \quad o_l = \begin{cases} n & \text{if } l = 0 \\ o_{l-1}/n_l, & \text{otherwise} \end{cases}, \tag{2.3}$$

where b_l and o_l denote number and order of blocks at level $l = 0, 1, \ldots, H$, respectively. Unrolling the recurrences, we obtain

$$b_l = \prod_{h=1}^{l} n_h^2 \quad \text{and} \quad o_l = \prod_{h=l+1}^{H} n_h \, .$$

Note that at level l there are $\sqrt{b_l}$ blocks each of order o_l along the diagonal of Q. Block $((i_1, \ldots, i_l), (j_1, \ldots, j_l))$ of Q at level l is given by

$$\begin{aligned} & Q((i_1, \ldots, i_l), (j_1, \ldots, j_l)) \\ &= \sum_{k=1}^{H+2K} \left(\prod_{h=1}^{l} q_k^{(h)}(i_h, j_h) \right) \left(\bigotimes_{h=l+1}^{H} Q_k^{(h)} \right) \quad \text{for } l = 0, \ldots, H \, , \end{aligned} \tag{2.4}$$

with the understanding that $l = 0$ yields Q and $l = H$ yields the scalar

$$q((i_1, \ldots, i_H), (j_1, \ldots, j_H)) = \sum_{k=1}^{H+2K} \prod_{h=1}^{H} q_k^{(h)}(i_h, j_h)$$

The nested structure associated with (2.2) is also valid in the presence of functional transitions. Now, let us consider a slightly different example.

Example 2. The model of a processing node with three submodels is depicted in Figure 2.2. The first submodel corresponds to the two processing elements (PEs), one acting as a cold spare, the second submodel corresponds to the bus, and the third submodel corresponds to the two memory elements (MEs). A processing node is up and running as long as one PE can access one ME through the bus, implying the PE, the bus, and at least one ME must be working for the model to be available. If the model is unavailable, no other components from that model can fail. Furthermore, there is one repairman in each submodel who can repair a failed component independently of other submodels. Time to failure is exponentially distributed with rate λ_1 for PEs, λ_2 for buses, and λ_3 for MEs. Repair times of components are also exponentially distributed. Repair rates of PEs, bus, and MEs are given, respectively, as μ_1, μ_2, and μ_3.

Letting $\mathbf{i} = (i_1, i_2, i_3)$ be the state representation of the model with i_1, i_2, and i_3, respectively, denoting the number of failed PEs, bus, and MEs, we have the 12 reachable states $(0,0,0)$, $(0,0,1)$, $(0,0,2)$, $(0,1,0)$, $(0,1,1)$, $(1,0,0)$, $(1,0,1)$, $(1,0,2)$, $(1,1,0)$, $(1,1,1)$, $(2,0,0)$, and $(2,0,1)$ out of the $3 \times 2 \times 3 = 18$ states in the product state space of the model. Note that

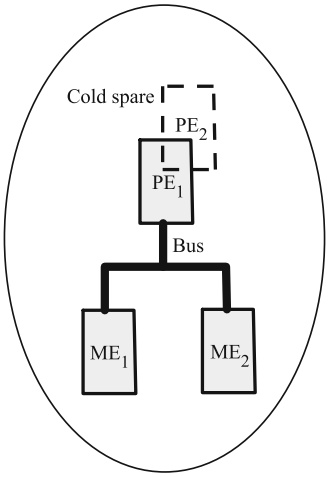


Fig. 2.2 Another availability model with three submodels

among the 12 reachable states, the model is available in only 4 of them, namely, $(0,0,0)$, $(0,0,1)$, $(1,0,0)$, and $(1,0,1)$.

By employing an automaton for each type of component, we can have a SAN model with three automata and no synchronizing transitions (i.e., $H = 3, K = 0$) yet with some local transitions that are functional. $\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$, and $\mathcal{A}^{(3)}$ model, respectively, PEs, bus, and MEs with state space sizes of $n_1 = 3$, $n_2 = 2$, and $n_3 = 3$ as in Figure 2.3.

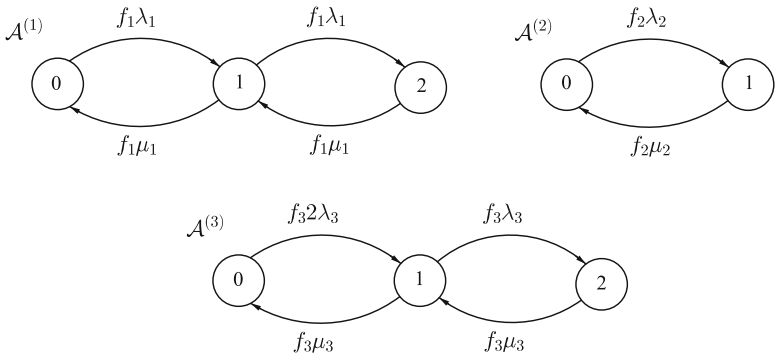


Fig. 2.3 SAN for availability model with three submodels

Note that the 2 in the rate of the transition from state 0 to state 1 in $\mathcal{A}^{(3)}$ is there because of the existence of two intact MEs that are competing with each other for failure. This is different than the rate of the transition from state 0 to state 1 in $\mathcal{A}^{(1)}$ where there is only PE that is working; the other PE is a cold spare and, therefore, not working and cannot fail.

The local transition rate matrices are given by

$$Q_l^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} * & f_1\lambda_1 & \\ f_1\mu_1 & * & f_1\lambda_1 \\ & f_1\mu_1 & * \end{pmatrix} \end{array}, \quad Q_l^{(2)} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} * & f_2\lambda_2 \\ f_2\mu_2 & * \end{pmatrix} \end{array},$$

$$Q_l^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} * & f_3 2\lambda_3 & \\ f_3\mu_3 & * & f_3\lambda_3 \\ & f_3\mu_3 & * \end{pmatrix} \end{array},$$

where

$$f_1 = \begin{cases} 1 & \text{if } i_2 = 0 \text{ and } i_3 < 2 \\ 0 & \text{otherwise} \end{cases}, \quad f_2 = \begin{cases} 1 & \text{if } i_1 < 2 \text{ and } i_3 < 2 \\ 0 & \text{otherwise} \end{cases},$$

$$\text{and} \quad f_3 = \begin{cases} 1 & \text{if } i_1 < 2 \text{ and } i_2 = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Although there are no explicit synchronizing transitions in this SAN, in fact each local transition in an automaton depends on the state of the other two automata through the values of the functions f_1, f_2, and f_3. Hence, we have generalized Kronecker products/sums. It is possible to express this explicitly, for instance, by placing a lowercase g (for generalized) as a subscript of the respective operator [136, 137].

The descriptor in (2.1) can be written as

$$Q = \bigoplus_{h=1}^{3} Q_l^{(h)} = Q_l^{(1)} \otimes I_6 + I_3 \otimes Q_l^{(2)} \otimes I_3 + I_6 \otimes Q_l^{(3)}.$$

The sum of the three Kronecker product terms above yields the flat generator matrix

$$Q = \begin{array}{c} \\ 0\,0\,0 \\ 0\,0\,1 \\ 0\,0\,2 \\ 0\,1\,0 \\ 0\,1\,1 \\ 0\,1\,2 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,0\,2 \\ 1\,1\,0 \\ 1\,1\,1 \\ 1\,1\,2 \\ 2\,0\,0 \\ 2\,0\,1 \\ 2\,0\,2 \\ 2\,1\,0 \\ 2\,1\,1 \\ 2\,1\,2 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{array} \\ \begin{pmatrix} * & 2\lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & & \\ \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & & & & & & \\ & \mu_3 & * & & & & & & & & & & & & & & & \\ \mu_2 & & & * & & & & & & & & & & & & & & \\ & \mu_2 & & \mu_3 & * & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & & & \\ \mu_1 & & & & & & * & 2\lambda_3 & & \lambda_2 & & & \lambda_1 & & & & & \\ & \mu_1 & & & & & \mu_3 & * & \lambda_3 & & \lambda_2 & & & \lambda_1 & & & & \\ & & \mu_1 & & & & & \mu_3 & * & & & & & & & & & \\ & & & \mu_1 & & & \mu_2 & & & * & & & & & & & & \\ & & & & \mu_1 & & & \mu_2 & & \mu_3 & * & & & & & & & \\ & & & & & & & & & & & & & & & & & \\ & & & & & & \mu_1 & & & & & & * & & & & & \\ & & & & & & & \mu_1 & & & & & \mu_3 & * & & & & \\ & & & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & & & \end{pmatrix} \end{array}.$$

The number of floating-point values stored in the SAN representation of Q is 18, whereas it is 42 for its flat representation. Furthermore, the SAN specification has to somehow code the functions f_1, f_2, and f_3 and provide a mechanism to evaluate them.

Observe that the rows and columns corresponding to the unreachable states (0,1,2), (1,1,2), (2,0,2), (2,1,0), (2,1,1), and (2,1,2) are zero. The unreachable states are those for which there are at least two failed submodels, whereas the operational semantics dictated there be at most one failed submodel and there could be no further failures once there is a failed submodel. Similarly, there cannot be repairs in states that are unreachable.

See [134] for a recent application of the SAN modeling approach in the presence of unreachable states to the analysis of a production line with multiple stations in tandem. The second modeling approach we discuss next does not represent unreachable states.

2.2 Hierarchical Markovian Models

Hierarchical Markovian models (HMMs) are composed of multiple *low-level models* (LLMs) and a *high-level model* (HLM) that defines how LLMs interact. States of the HLM are called *macrostates*, whereas states defined by LLMs are called *microstates*. When there is only one macrostate, the HMM representation is equivalent to a SAN representation that does not have any unreachable states. In this representation, each LLM may be viewed as an automaton. HMMs with multiple macrostates are used to specify models that have unreachable states in their product state spaces. Formal treatment of HMMs can be found, for instance, in [45]. Even though the definition of HMMs can be generalized to more than two model levels, we will be using the hierarchical structuring suggested in [49] which has one HLM and multiple LLMs.

In a given HMM with N macrostates numbered 0 through $N-1$, let H be the number of LLMs, let $\mathcal{R}_p^{(h)}$ be the subset of LLM h's state space $\mathcal{S}^{(h)}$ mapped to macrostate p (i.e., $\mathcal{R}_p^{(h)} \subseteq \mathcal{S}^{(h)}$), let $\mathcal{T}_{p,w}$ be the set of LLM (when $p = w$, nonlocal) transitions in the (p, w)th entry of the HLM matrix, let α_{t_e} be the rate associated with transition $t_e \in \mathcal{T}_{p,w}$, and let $D(p)$ be the diagonal (correction) matrix that sums the rows of Q corresponding to macrostate p to zero for $p = 0, \ldots, N-1$.

The meaning of these definitions will become clear as we progress into this section. In particular, we will see that an HMM provides a Kronecker representation for a model in which unreachable states that exist in its product state space are omitted, N is equal to the number of partitions of the resulting reachable state space, and each macrostate corresponds to a different reachable state space partition. Hence, the HLM in an HMM represents transitions among macrostates, in other words, reachable state space partitions of the model. Note that when $N = 1$, p becomes 0 and $\mathcal{R}_p^{(h)} = \mathcal{S}^{(h)}$ for $h = 1, \ldots, H$ as in SANs.

The HMM therefore defines an $(N \times N)$ block matrix

$$Q = \begin{pmatrix} Q(0,0) & \dots & Q(0,N-1) \\ \vdots & \ddots & \vdots \\ Q(N-1,0) & \dots & Q(N-1,N-1) \end{pmatrix}$$

associated with the *HLM matrix*

$$\begin{pmatrix} \mathcal{T}_{0,0} & \dots & \mathcal{T}_{0,N-1} \\ \vdots & \ddots & \vdots \\ \mathcal{T}_{N-1,0} & \dots & \mathcal{T}_{N-1,N-1} \end{pmatrix}$$

and H LLMs. Diagonal block (p,p) of Q corresponding to the (p,p)th entry of the HLM matrix is given by

$$Q(p,p) = \bigoplus_{h=1}^{H} Q_{t_0}^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) + \sum_{t_e \in \mathcal{T}_{p,p}} \alpha_{t_e} \bigotimes_{h=1}^{H} Q_{t_e}^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) + D(p), \quad (2.5)$$

where t_0 is used to denote local transitions. When $N = 1$, this representation is identical to that of the SAN descriptor in (2.1). The exception is that the transition rate α_{t_e} in the second term appears in front of the Kronecker product rather than being multiplied with the transition matrix of the master automaton in SANs and therefore hidden.

When there are multiple macrostates (i.e., $N > 1$), the off-diagonal block (p,w) of Q corresponding to the (p,w)th entry of the HLM matrix for $p, w = 0, \dots, N-1$ and $p \neq w$ is given by

$$Q(p,w) = \sum_{t_e \in \mathcal{T}_{p,w}} \alpha_{t_e} \bigotimes_{h=1}^{H} Q_{t_e}^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)}) \,. \quad (2.6)$$

By moving the rates α_{t_e} outside the transition matrices, the HMM representation circumvents having to identify one LLM as master and the other LLMs as slaves of transition t_e. The transition rates are scalars that multiply the corresponding Kronecker products.

When there are multiple macrostates, Q is a block matrix having as many blocks in each dimension as the number of macrostates (i.e., order of the HLM matrix). The diagonal and off-diagonal blocks of this partitioning are, respectively, the $Q(p,p)$ and $Q(p,w)$ matrices defined by (2.5) and (2.6). The diagonal of Q is formed of its negated off-diagonal row sums and may be stored explicitly as a vector or can be generated from sums of Kronecker products using the identity [61]

$$D(p) = -\bigoplus_{h=1}^{H} \mathrm{diag}(Q_{t_0}^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})\mathbf{e}) - \sum_{w=1}^{N-1} \sum_{t_e \in \mathcal{T}_{p,w}} \alpha_{t_e} \bigotimes_{h=1}^{H} \mathrm{diag}(Q_{t_e}^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}) \tag{2.7}$$

for $p = 0, \ldots, N-1$ when needed.

Macrostates in an HLM may have different numbers of microstates when LLMs have partitioned state spaces. The microstates corresponding to each macrostate result from the Cartesian product of the state space partitions of LLMs that are mapped to that particular macrostate without unreachable states. This is something we will explain in detail through the examples in this chapter but also investigate further in Chapter 3.

Now, let us show how we can represent the availability model with three submodels and 12 reachable states in Example 2 as an HMM.

Example 2. (ctnd.) First we need to specify a partitioning for the *reachable state space*, $\mathcal{R}$, of the model. Given $\mathcal{S}^{(1)} = \mathcal{S}^{(3)} = \{0, 1, 2\}$ and $\mathcal{S}^{(2)} = \{0, 1\}$ as the state spaces of the LLMs, and

$$\mathcal{R} = \{(0,0,0), (0,0,1), (0,0,2), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,0,2), (1,1,0), (1,1,1), (2,0,0), (2,0,1)\} ,$$

we remark that $|\mathcal{R}| = 12$, whereas $|\mathcal{S}| = 18$, hence, $\mathcal{R} \subset \mathcal{S}$, where $\mathcal{S} = \bigtimes_{h=1}^{3} \mathcal{S}^{(h)}$ is the product state space.

As it will soon become clear, a suitable partitioning seems to be

$$\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3 ,$$

where

$$\mathcal{R}_0 = \{(0,0,0), (0,0,1), (1,0,0), (1,0,1)\} , \quad \mathcal{R}_1 = \{(2,0,0), (2,0,1)\} ,$$

$$\mathcal{R}_2 = \{(0,1,0), (0,1,1), (1,1,0), (1,1,1)\} , \quad \mathcal{R}_3 = \{(0,0,2), (1,0,2)\} .$$

Note that $\mathcal{R}_0$ corresponds to those states in which the model is available, whereas $\mathcal{R}_p$ corresponds to those states in which the model is unavailable and this unavailability is due a failed submodel $p = 1, 2, 3$.

We can express the reachable state space partitions above as

$$\mathcal{R}_0 = \bigtimes_{h=1}^{3} \mathcal{R}_0^{(h)} , \quad \mathcal{R}_1 = \bigtimes_{h=1}^{3} \mathcal{R}_1^{(h)} , \quad \mathcal{R}_2 = \bigtimes_{h=1}^{3} \mathcal{R}_2^{(h)} , \quad \mathcal{R}_3 = \bigtimes_{h=1}^{3} \mathcal{R}_3^{(h)} ,$$

where

$$\mathcal{R}_0^{(1)} = \mathcal{R}_2^{(1)} = \mathcal{R}_3^{(1)} = \mathcal{R}_0^{(3)} = \mathcal{R}_1^{(3)} = \mathcal{R}_2^{(3)} = \{0, 1\} , \quad \mathcal{R}_1^{(1)} = \mathcal{R}_3^{(3)} = \{2\} ,$$

$$\mathcal{R}_0^{(2)} = \mathcal{R}_1^{(2)} = \mathcal{R}_3^{(2)} = \{0\}\,, \quad \text{and} \quad \mathcal{R}_2^{(2)} = \{1\}$$

Hence, we contemplate $N = 4$ macrostates.

The mapping between HLM states and LLM states dictated by the partitioning of LLM state spaces is shown in Table 2.1.

Table 2.1 Mapping between HLM and LLM states in availability model with three submodels

HLM	LLM 1	LLM 2	LLM 3	# of microstates
0	0:1	0:0	0:1	2 . 1 . 2 = 4
1	2:2	0:0	0:1	1 . 1 . 2 = 2
2	0:1	1:1	0:1	2 . 1 . 2 = 4
3	0:1	0:0	2:2	2 . 1 . 1 = 2

In the particular model under consideration, six transitions denoted by t_1 through t_6 take place in the HLM and affect the LLMs. These transitions are captured by the (4×4) HLM matrix

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} \{t_1, t_3, t_4, t_6\} & \{t_1\} & \{t_2\} & \{t_3\} \\ \{t_4\} & \{t_6\} & & \\ \{t_5\} & & \{t_4, t_6\} & \\ \{t_6\} & & & \{t_4\} \end{pmatrix} \end{array}. \tag{2.8}$$

Transitions t_1 through t_3 represent the failure of a working component, while t_4 through t_6 represent the repair of a failed component in LLMs 1 through 3, respectively.

Other than Kronecker products due to the depicted transitions in (2.8), there is a Kronecker sum implicitly associated with each diagonal entry of the HLM matrix. Each Kronecker sum is formed of three LLM matrices corresponding to local transition t_0. The matrices $Q_{t_0}^{(1)}$, $Q_{t_0}^{(2)}$, $Q_{t_0}^{(3)}$ associated with local transitions are all 0 in this HMM.

To each transition t_1 through t_6 corresponds a Kronecker product of three LLM matrices. The matrices associated with those LLMs that do not participate in a transition are all identity. LLM 1 participates in t_1 and t_4 with the matrices $Q_{t_1}^{(1)}$ and $Q_{t_4}^{(1)}$; LLM 2 participates in t_2 and t_5 with the matrices $Q_{t_2}^{(2)}$ and $Q_{t_5}^{(2)}$; and LLM 3 participates in t_3 and t_6 with the matrices $Q_{t_3}^{(3)}$ and $Q_{t_6}^{(3)}$.

The LLM matrices associated with transitions t_1 through t_6 are

$$Q_{t_1}^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left(\begin{array}{cc|c} & 1 & \\ & & \\ \hline & & 1 \end{array} \right) \end{array}, \quad Q_{t_2}^{(2)} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \left(\begin{array}{c|c} & 1 \\ \hline & \end{array} \right) \end{array}, \quad Q_{t_3}^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left(\begin{array}{cc|c} & 2 & \\ & & \\ \hline & & 1 \end{array} \right) \end{array},$$

$$Q_{t_4}^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left(\begin{array}{cc|c} & & \\ 1 & & \\ \hline & 1 & \end{array} \right) \end{array}, \quad Q_{t_5}^{(2)} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \left(\begin{array}{c|c} & \\ 1 & \\ \hline \end{array} \right) \end{array}, \quad Q_{t_6}^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \left(\begin{array}{cc|c} & & \\ 1 & & \\ \hline & 1 & \end{array} \right) \end{array},$$

$$Q_{t_2}^{(1)} = Q_{t_3}^{(1)} = Q_{t_5}^{(1)} = Q_{t_6}^{(1)} = Q_{t_1}^{(3)} = Q_{t_2}^{(3)} = Q_{t_4}^{(3)} = Q_{t_5}^{(3)} = I_3 \,,$$

$$Q_{t_1}^{(2)} = Q_{t_3}^{(2)} = Q_{t_4}^{(2)} = Q_{t_6}^{(2)} = I_2 \,.$$

The rates of the transitions are given by

$$(\alpha_{t_1}, \alpha_{t_2}, \alpha_{t_3}, \alpha_{t_4}, \alpha_{t_5}, \alpha_{t_6}) = (\lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2, \mu_3) \,.$$

In this model, each of the transitions t_1 through t_6 affects only one LLM. For instance, the Kronecker product associated with t_3 in entry (0,0) of the HLM matrix in (2.8) is

$$I_2 \otimes I_1 \otimes Q_{t_3}^{(3)}(0:1,0:1) = \begin{array}{c} \\ 0\,0\,0 \\ 0\,0\,1 \\ 1\,0\,0 \\ 1\,0\,1 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{array} \\ \begin{pmatrix} & 2 & & \\ & & & \\ & & & 2 \\ & & & \end{pmatrix} \end{array},$$

where $Q_{t_3}^{(3)}(0:1,0:1)$ denotes the submatrix of $Q_{t_2}^{(3)}$ that lies between states 0 through 1 rowwise and states 0 through 1 columnwise. Obviously, this Kronecker product gets multiplied with the rate of transition t_3, that is, λ_3. On the other hand, the Kronecker product associated with t_4 in entry (1,0) of the HLM matrix in (2.8) is

$$Q_{t_4}^{(1)}(2:2,0:1) \otimes I_1 \otimes I_2 = \begin{array}{c} \\ 2\,0\,0 \\ 2\,0\,1 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{array} \\ \begin{pmatrix} & & 1 & \\ & & & 1 \end{pmatrix} \end{array}.$$

This Kronecker product gets multiplied with the rate of transition t_4, that is, μ_1.

The Kronecker representation associated with the HMM needs to store 1 HLM matrix having 6 floating-point values for the rates of transitions t_1 through t_6 and 6 LLM matrices (since identity matrices are not stored) having

a total of 10 floating-point values, thus altogether 16 floating-point values. If the diagonal entries in $D(p)$ for $p = 0, \ldots, 3$ in (2.5) are stored explicitly, an additional 12 floating-point values will be required. The underlying generator matrix is given by

$$Q = \begin{array}{c} \\ 0\,0\,0 \\ 0\,0\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 2\,0\,0 \\ 2\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,1\,0 \\ 1\,1\,1 \\ 0\,0\,2 \\ 1\,0\,2 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 2 \end{array} \\ \left(\begin{array}{cccc|cc|cccc|cc} * & 2\lambda_3 & \lambda_1 & & & & \lambda_2 & & & & & \\ \mu_3 & * & & \lambda_1 & & & & \lambda_2 & & & \lambda_3 & \\ \mu_1 & & * & 2\lambda_3 & \lambda_1 & & & & \lambda_2 & & & \\ & \mu_1 & \mu_3 & * & & \lambda_1 & & & & \lambda_2 & & \lambda_3 \\ \hline & & \mu_1 & & * & & & & & & & \\ & & & \mu_1 & \mu_3 & * & & & & & & \\ \hline \mu_2 & & & & & & * & & & & & \\ & \mu_2 & & & & & \mu_3 & * & & & & \\ & & \mu_2 & & & & \mu_1 & & * & & & \\ & & & \mu_2 & & & & \mu_1 & \mu_3 & * & & \\ \hline & \mu_3 & & & & & & & & & * & \\ & & & \mu_3 & & & & & & & \mu_1 & * \end{array}\right) \end{array} . \tag{2.9}$$

As in SANs, the Kronecker structure of an HMM induces nested block partitionings in the generator matrix underlying the CTMC. Assuming that the HLM has multiple states, it suggests the block partitioning at level 0. The LLMs define the nested block partitionings at higher level numbers. LLM 1 defines the block partitioning at level 1. LLMs 1 and 2 define the block partitioning at level 2, and so on. In HMMs, diagonal blocks at a particular level of the nested block partitioning are all square but can have different orders in different HLM states. Consequently, blocks in the off-diagonal part of the HLM matrix need not be square. This is different than SANs in which all (diagonal and off-diagonal) blocks at each level of the nested block partitioning associated with the Kronecker structure are square and have the same order.

Example 2. (ctnd.) In our HMM, the HLM matrix which has 4 macrostates defines the block partitioning of Q in (2.9) at level 0. Along the diagonal of Q, there are 4 blocks which are, respectively, (4×4), (2×2), (4×4), and (2×2). LLM 1 defines the block partitioning at level 1 as in

$$Q = \left(\begin{array}{cc|cc|cc|cc|cc|cc|cc|cc}
* & 2\lambda_3 & \lambda_1 & & & & \lambda_2 & & & & & & & \\
\mu_3 & * & & \lambda_1 & & & & \lambda_2 & & & \lambda_3 & & & \\
\hline
\mu_1 & & * & 2\lambda_3 & \lambda_1 & & & & \lambda_2 & & & & & \\
& \mu_1 & \mu_3 & * & & \lambda_1 & & & & \lambda_2 & & \lambda_3 & & \\
\hline
& & \mu_1 & & * & & & & & & & & & \\
& & & \mu_1 & \mu_3 & * & & & & & & & & \\
\hline
\mu_2 & & & & & & * & & & & & & & \\
& \mu_2 & & & & & \mu_3 & * & & & & & & \\
\hline
& & \mu_2 & & & & \mu_1 & & * & & & & & \\
& & & \mu_2 & & & & \mu_1 & \mu_3 & * & & & & \\
\hline
& \mu_3 & & & & & & & & & * & & & \\
& & & \mu_3 & & & & & & & \mu_1 & * & & \\
\end{array}\right).$$

Since LLM 2 contributes only a single state to each macrostate, the block partitioning of Q at level 2 for our HMM is the same as its block partitioning at level 1. Finally, the block partitioning of Q at level 3 due to LLMs 1 through 3 for our HMM is given by

$$Q = \left(\begin{array}{cc|cc|cc|cc|cc|cc}
* & 2\lambda_3 & \lambda_1 & & & & \lambda_2 & & & & & \\
\mu_3 & * & & \lambda_1 & & & & \lambda_2 & & & \lambda_3 & \\
\hline
\mu_1 & & * & 2\lambda_3 & \lambda_1 & & & & \lambda_2 & & & \\
& \mu_1 & \mu_3 & * & & \lambda_1 & & & & \lambda_2 & & \lambda_3 \\
\hline
& & \mu_1 & & * & & & & & & & \\
& & & \mu_1 & \mu_3 & * & & & & & & \\
\hline
\mu_2 & & & & & & * & & & & & \\
& \mu_2 & & & & & \mu_3 & * & & & & \\
\hline
& & \mu_2 & & & & \mu_1 & & * & & & \\
& & & \mu_2 & & & & \mu_1 & \mu_3 & * & & \\
\hline
& \mu_3 & & & & & & & & & * & \\
& & & \mu_3 & & & & & & & \mu_1 & * \\
\end{array}\right).$$

Now, let us replace the Kronecker sum with a sum of Konecker products as we have done for SANs, include the local transitions along the diagonal of the HLM matrix, number local transitions from 1 through H, number synchronizing transitions starting from $H + 1$, and make the subscripts in the transitions the transition numbers themselves to simplify the notation. Then the model defines an $(N \times N)$ block matrix

$$Q = \begin{pmatrix} Q(0,0) & \dots & Q(0,N-1) \\ \vdots & \ddots & \vdots \\ Q(N-1,0) & \dots & Q(N-1,N-1) \end{pmatrix} \tag{2.10}$$

associated with the *interaction matrix*

$$\begin{pmatrix} \mathcal{K}_{0,0} & \cdots & \mathcal{K}_{0,N-1} \\ \vdots & \ddots & \vdots \\ \mathcal{K}_{N-1,0} & \cdots & \mathcal{K}_{N-1,N-1} \end{pmatrix} \tag{2.11}$$

and H submodels with state spaces $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$. Block (p, w) of Q for $p, w = 0, \ldots, N-1$ is expressed as

$$Q(p,w) = \begin{cases} \sum_{k \in \mathcal{K}_{p,w}} Q_k(p,w) + Q_D(p,p) & \text{if } p = w \\ \sum_{k \in \mathcal{K}_{p,w}} Q_k(p,w) & \text{otherwise} \end{cases},$$

where

$$Q_k(p,w) = \alpha_k \bigotimes_{h=1}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)}),$$

$$Q_D(p,p) = -\sum_{w=0}^{N-1} \sum_{k \in \mathcal{K}_{p,w}} \alpha_k \bigotimes_{h=1}^{H} \mathrm{diag}(Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}),$$

α_k is the rate associated with transition k in block (p, w), $\mathcal{K}_{p,w}$ is the set of transitions in block (p, w), and $Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$ is the submatrix of the *transition matrix* $Q_k^{(h)}$ whose row and column state spaces are $\mathcal{R}_p^{(h)} \subseteq \mathcal{S}^{(h)}$ and $\mathcal{R}_w^{(h)} \subseteq \mathcal{S}^{(h)}$, respectively. Hence,

$$\mathcal{R}_p = \bigtimes_{h=1}^{H} \mathcal{R}_p^{(h)} \quad \text{and} \quad \mathcal{R}_w = \bigtimes_{h=1}^{H} \mathcal{R}_w^{(h)}$$

are the reachable state space partitions corresponding to p and w, respectively. Note that the rate α_k of the Kronecker product term $Q_k(p, w)$ can be eliminated by scaling one of the $Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$ matrices in the term with that rate as is done for the master automaton in SANs. The case of $N = 1$ implies $\mathcal{R} = \mathcal{S}$, meaning no unreachable states. We let

$$\mathcal{K} = \bigcup_{p,w=0}^{N-1} \mathcal{K}_{p,w}$$

denote the *set of transitions* and $K = |\mathcal{K}|$ denote the number of transitions in the model.

2.3 Two Kronecker-Structured Models

In this section, we consider a relatively complicated availability model and a relatively complicated polling model each having H submodels. In coming up

with suitable specifications for these two models, we develop the Kronecker-based representation to be used throughout this book. Furthermore, these models later serve as benchmarks for analysis.

2.3.1 An Availability Model

We motivate the discussion with the availability model of H submodels in Figure 2.4 (see also [61, 64]). This is a model in which each submodel represents a processing node with two PEs, one acting as a cold spare, a bus, and two MEs as in Figure 2.2. Here, we choose to represent the processing node as

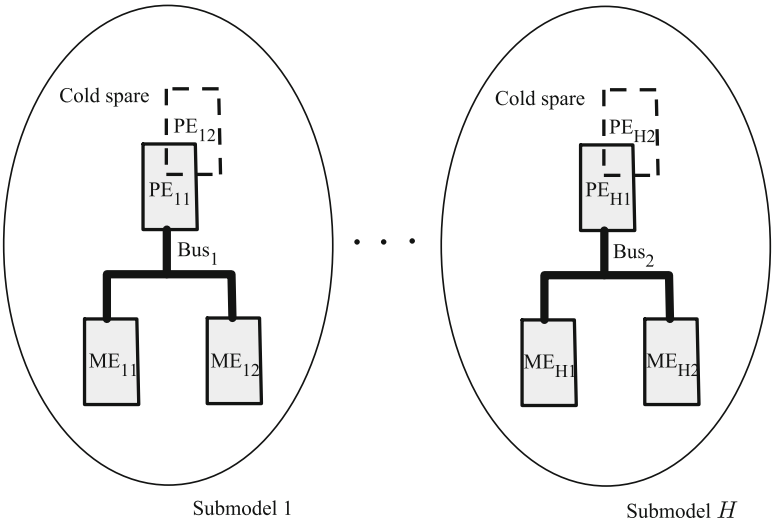


Fig. 2.4 Availability model with H submodels

a single submodel with the 12 reachable states described in Example 2. This is called *grouping* (since we have grouped the three submodels forming the processing node into a single submodel) and will be discussed in more detail in Chapter 4. Hence, we have a reachable state space equal to the product state space containing 12^H states for this H-dimensional availability model.

Recall that a processing node is up and running as long as one PE can access one ME through the bus, implying the PE, the bus, and at least one ME must be working for the submodel to be available. If the submodel is unavailable, no other components from that submodel can fail. Submodels are repaired by a global repair facility with preemptive priority such that submodel 1 has the highest priority and submodel H has the least priority. In each submodel, the repair of a failed component takes place independently of other components in the same submodel.

Time to failure is exponentially distributed with rate $\lambda_1^{(h)}$ for PEs, $\lambda_2^{(h)}$ for the bus, and $\lambda_3^{(h)}$ for MEs in submodel h. Repair times of components are also exponentially distributed. Repair rates of PEs, the bus, and MEs in submodel h are given, respectively, as $\mu_1^{(h)}$, $\mu_2^{(h)}$, and $\mu_3^{(h)}$. Note that the availability model cannot be considered as being symmetric due to the existence of the priority repair strategy among submodels. Furthermore, as it is common in availability models, it incorporates different time scales, and hence, the steady-state probability distribution becomes unbalanced due to the repair rates being much larger than failure rates.

We will be considering models with $H = 3, 4, 5, 6, 7, 8$. Here, we provide a Kronecker representation of this model for $H = 3$. The states of each submodel are numbered 0 through 11, and the submodel state spaces are given by $\mathcal{S}^{(h)} = \{0, \ldots, 11\}$ for $h = 1, 2, 3$. There are altogether $K = 9$ transitions numbered 1 through 9 in the model, the first three of which may be perceived as being local.

Transition 1 models the self evolution of submodel 1, which has the highest priority among the three submodels, with the matrices

$$Q_1^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\ \begin{pmatrix} & 2\lambda_3^{(1)} & \lambda_1^{(1)} & & & & \lambda_2^{(1)} & & & & & \\ \mu_3^{(1)} & & & \lambda_1^{(1)} & & & & \lambda_2^{(1)} & & & \lambda_3^{(1)} & \\ \mu_1^{(1)} & & & 2\lambda_3^{(1)} & \lambda_1^{(1)} & & & & \lambda_2^{(1)} & & & \\ & \mu_1^{(1)} & \mu_3^{(1)} & & & \lambda_1^{(1)} & & & & \lambda_2^{(1)} & & \lambda_3^{(1)} \\ & & \mu_1^{(1)} & & & & & & & & & \\ & & & \mu_1^{(1)} & \mu_3^{(1)} & & & & & & & \\ \mu_2^{(1)} & & & & & & & & & & & \\ & \mu_2^{(1)} & & & & & \mu_3^{(1)} & & & & & \\ & & \mu_2^{(1)} & & & & \mu_1^{(1)} & & & & & \\ & & & \mu_2^{(1)} & & & & \mu_1^{(1)} & \mu_3^{(1)} & & & \\ & \mu_3^{(1)} & & & & & & & & & & \\ & & & \mu_3^{(1)} & & & & & & & \mu_1^{(1)} & \end{pmatrix} \end{array},$$

$$Q_1^{(2)} = Q_1^{(3)} = I_{12}.$$

Note that transitions due to failures appear in the upper-triangular part, whereas transitions due to repairs appear in the lower-triangular part of $Q_1^{(1)}$ for which $nz_{Q_1^{(1)}} = 30$. Because of their priority, repairs in submodel 1 do not depend on the states of submodels 2 and 3. This is not the case for submodels 2 and 3. Hence, we need to represent the repairs within submodels 2 and 3 separately from failures.

Transitions 2 and 3 represent the failure behavior of submodels 2 and 3, respectively, and therefore, the corresponding matrices have their nonzeros

in exactly the same entries as in the upper-triangular part of $Q_1^{(1)}$. That is, for $h = 2, 3$ we have

$$Q_h^{(h)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\ \begin{pmatrix} & 2\lambda_3^{(h)} & \lambda_1^{(h)} & & & & \lambda_2^{(h)} & & & & & \\ & & & \lambda_1^{(h)} & & & & \lambda_2^{(h)} & & & \lambda_3^{(h)} & \\ & & & 2\lambda_3^{(h)} & \lambda_1^{(h)} & & & & \lambda_2^{(h)} & & & \\ & & & & & \lambda_1^{(h)} & & & & \lambda_2^{(h)} & & \lambda_3^{(h)} \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \\ & & & & & & & & & & & \end{pmatrix} \end{array},$$

$$Q_2^{(1)} = Q_2^{(3)} = Q_3^{(1)} = Q_3^{(2)} = I_{12} \ .$$

Note that we have $nz_{Q_2^{(2)}} = nz_{Q_3^{(3)}} = 12$, and the 12 nonzeros are located in the first four rows of each matrix.

Let us for a moment assume that the repair of the bus had priority over the repair of the PE which had priority over the repair of the ME. Then the seven transitions in entries $(3, 2)$, $(5, 4)$, $(7, 6)$, $(8, 6)$, $(9, 7)$, $(9, 8)$, and $(11, 3)$ due to repairs in each submodel would need to be inhibited.

We use transitions 4 through 9, altogether 6 transitions, to represent repairs in submodels 2 and 3. Transitions 4 through 6 are associated with repairs in submodel 2 and transitions 7 through 9 are associated with repairs in submodel 3. Observe that repairs in submodels 2 and 3 are only possible when submodel 1 is in state 0, that is, when all its components are intact. Therefore, submodel 1 has the (12×12) rank–1 inhibition matrices

$$Q_4^{(1)} = Q_5^{(1)} = Q_6^{(1)} = Q_7^{(1)} = Q_8^{(1)} = Q_9^{(1)} = \mathbf{e}_0 \mathbf{e}_0^T$$

associated with transitions 4 through 9. Recall that $\mathbf{e}_0$ represents a column vector with a 1 in its 0th entry and all its remaining entries being 0. Hence, $\mathbf{e}_0 \mathbf{e}_0^T$ is the matrix having a 1 in its (0,0)th entry and 0s elsewhere, implying that these 6 transitions can only take place when submodel 1 is in state 0, and once these transitions are executed submodel 1 remains in state 0. Similarly, since submodel 2 has priority over submodel 3 in repairs, submodel 2 has the (12×12) rank–1 inhibition matrices

$$Q_7^{(2)} = Q_8^{(2)} = Q_9^{(2)} = \mathbf{e}_0 \mathbf{e}_0^T$$

associated with transitions 7 through 9. These matrices imply that transitions 7, 8, and 9 can only take place when submodel 2 is in state 0, and once they take place submodel 2 remains in state 0.

Transitions 4 through 6 represent repairs in submodel 2 when submodel 1 is in state 0. We let transition 4 represent the repair of PEs in submodel 2. Hence, we have

$$Q_4^{(2)} = \mathbf{e}_2\mathbf{e}_0^T + \mathbf{e}_3\mathbf{e}_1^T + \mathbf{e}_4\mathbf{e}_2^T + \mathbf{e}_5\mathbf{e}_3^T + \mathbf{e}_8\mathbf{e}_6^T + \mathbf{e}_9\mathbf{e}_7^T + \mathbf{e}_{11}\mathbf{e}_{10}^T \, .$$

as a sum of 7 rank–1 matrices whose nonzero values appear in the same entries where we have $\mu_1^{(1)}$ in $Q_1^{(1)}$. Similarly, we let transitions 5 and 6 represent the repairs of bus and MEs, respectively, and obtain

$$Q_5^{(2)} = \mathbf{e}_6\mathbf{e}_0^T + \mathbf{e}_7\mathbf{e}_1^T + \mathbf{e}_8\mathbf{e}_2^T + \mathbf{e}_9\mathbf{e}_3^T$$

as a sum of 4 rank–1 matrices, and

$$Q_6^{(2)} = \mathbf{e}_1\mathbf{e}_0^T + \mathbf{e}_3\mathbf{e}_2^T + \mathbf{e}_5\mathbf{e}_4^T + \mathbf{e}_7\mathbf{e}_6^T + \mathbf{e}_9\mathbf{e}_8^T + \mathbf{e}_{10}\mathbf{e}_1^T + \mathbf{e}_{11}\mathbf{e}_3^T$$

as a sum of 7 rank–1 matrices.

Transitions 7 through 9 represent repairs in submodel 3 when submodels 1 and 2 are in state 0. We let transitions 7, 8, and 9 represent the repairs of PEs, bus, and MEs in submodel 3, respectively. Hence, we have

$$Q_7^{(3)} = Q_4^{(2)}, \quad Q_8^{(3)} = Q_5^{(2)}, \quad \text{and} \quad Q_9^{(3)} = Q_6^{(2)} \, .$$

Furthermore, we have

$$Q_4^{(3)} = Q_5^{(3)} = Q_6^{(3)} = I_{12}$$

since submodel 3 does not influence in any way the repair of components in submodel 2. We remark that the triple matrices $Q_4^{(2)}$, $Q_5^{(2)}$, $Q_6^{(2)}$ have a total of 18 nonzeros. This is also the case for $Q_7^{(3)}$, $Q_8^{(3)}$, $Q_9^{(3)}$.

Finally, rates of transitions 1 through 9 are given by

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9) = (1, 1, 1, \mu_1^{(2)}, \mu_2^{(2)}, \mu_3^{(2)}, \mu_1^{(3)}, \mu_2^{(3)}, \mu_3^{(3)}) \, .$$

In general, there are a total of $H + 3(H-1)$ transitions. Assuming that the transitions are numbered as in $H = 3$, the first H transitions are local with rates of 1. The matrix $Q_1^{(1)}$ is the same as in $H = 3$. The $Q_k^{(k)}$ matrices corresponding to transitions $k = 2, \ldots, H$ all have the nonzero structure of $Q_2^{(2)}$ as in $H = 3$; all other matrices corresponding to transitions $k = 1, \ldots, H$ satisfy $Q_k^{(h)} = I_{12}$ for $h \neq k$, $h = 1, \ldots, H$. The remaining transitions numbered $H+3(k-1)+1$, $H+3(k-1)+2$, $H+3k$, respectively, have rates of $\mu_1^{(k+1)}$, $\mu_2^{(k+1)}$, $\mu_3^{(k+1)}$ for $k = 1, \ldots, H-1$. The (12×12) matrices associated

with these transitions for submodel h other than 1 satisfy $Q_k^{(h)} = I_{12}$ for $k = H+1, \ldots, 3h-3$, the triple matrices $Q_{3h-2}^{(h)}$, $Q_{3h-1}^{(h)}$, $Q_{3h}^{(h)}$, respectively, are equal to $Q_4^{(2)}$, $Q_5^{(2)}$, $Q_6^{(2)}$ as in $H = 3$, and the remaining matrices satisfy $Q_k^{(h)} = \mathbf{e}_0 \mathbf{e}_0^T$ for $k = 3h+1, \ldots, 4H-3$.

Kronecker representation of the underlying generator matrix for the availability model may then be written as

$$Q = \sum_{k=1}^{4H-3} \alpha_k \bigotimes_{h=1}^{H} Q_k^{(h)} + Q_D, \quad Q_D = -\sum_{k=1}^{4H-3} \alpha_k \bigotimes_{h=1}^{H} \mathrm{diag}(Q_k^{(h)} \mathbf{e}) \; .$$

The number of floating-point values to be stored in this representation is $4H-3$ for the rates, $30H$ for the values in the matrices representing the failures and repairs, and $3\sum_{h=1}^{H-1}(H-h) = 3(H-1)H/2$ for the rank-1 inhibition matrices, thus altogether $1.5H^2 + 28.5H$. For $H = 8$, this number becomes 324. On the other hand, the number of nonzeros in the off-diagonal part of Q is $30 \cdot 12^{H-1}$ due to transition 1, $12(H-1)12^{H-1}$ due to transitions 2 through H, and $18\sum_{h=2}^{H} 12^{H-h}$ due to the remaining transitions. For $H = 8$, Q is of order 429,981,696 and has a total of 4,143,459,978 nonzero off-diagonal entries.

2.3.2 A Polling Model

We consider a model of the multiserver multiqueue discussed in [3]. The model consists of H submodels each having a finite capacity queue at which customers arrive according to a Poisson process as in Figure 2.5 (see also [50, 58, 59, 60, 61, 64, 110]). Interarrival times of customers at the queue in submodel h are exponentially distributed with rate λ_h for $h = 1, \ldots, H$. Hence, probability of customer arrival at queue h from the aggregate Poisson arrival process with rate $\sum_{h=1}^{H} \lambda_h$ is obtained as $\lambda_h / \sum_{h=1}^{H} \lambda_h$. The capacity of queue h is equal to C_h. Arrivals at a full queue are assumed to get lost, and served customers are assumed to leave their submodels.

In this H-dimensional model, there are S servers serving (in other words, polling) the H queues in a round-robin manner. When a server arrives at a queue with customers, it serves the customer at the head of the queue and then travels to the next queue in line. On the other hand, a server that arrives at an empty queue skips the service phase and travels to the next queue in line. Service times at queue h and travel times from submodel h to submodel $1 + (h \bmod H)$ are exponentially distributed, respectively, with rates μ_h and γ_h. The modulus operator is used to take care of the wrap-around effect when in submodel H. Normally, we would have $S \geqslant 1$ and $C_h \geqslant S$ for $h = 1, \ldots, H$. Figure 2.5 depicts one server in submodel 1 serving the customer at the head of the queue with two customers and one server traveling from submodel H in which the queue is empty to submodel 1.

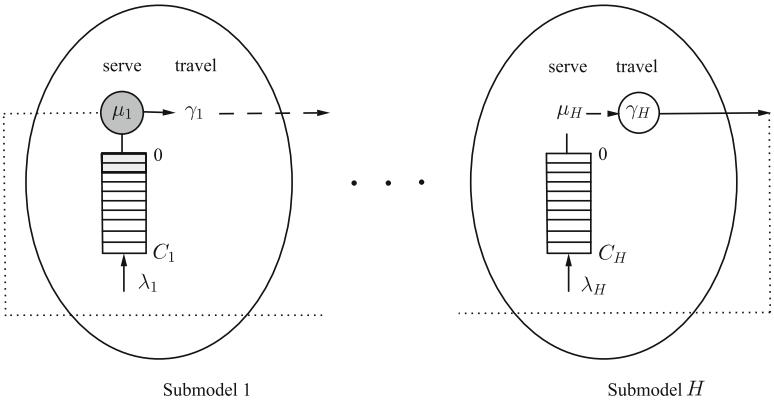


Fig. 2.5 Polling model with H submodels

Observe that S servers can be distributed among H queues in

$$N = \binom{S+H-1}{S}$$

different ways; there can be S servers simultaneously serving S different customers at queue h when $C_h \geqslant S$ or S servers simultaneously traveling from one submodel to the next. The constraint regarding the distribution of servers among submodels hints at a reachable state space, $\mathcal{R}$, that is a subset of the product state space, $\mathcal{S}$. This in turn implies a partitioning of the submodel state spaces $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$ similar to that in Example 2.

When $S = 2$, queue h with capacity C_h can be represented as part of submodel h with state space

$$\mathcal{S}^{(h)} = \mathcal{S}_0^{(h)} \cup \mathcal{S}_1^{(h)} \cup \mathcal{S}_2^{(h)} ,$$

where

$$\begin{aligned}
\mathcal{S}_0^{(h)} &= \{0, \ldots, C_h\}, \\
\mathcal{S}_1^{(h)} &= \{C_h + 1, \ldots, 2C_h + 1\} \cup \{2C_h + 2, \ldots, 3C_h + 1\}, \\
\mathcal{S}_2^{(h)} &= \{3C_h + 2, \ldots, 4C_h + 2\} \cup \{4C_h + 3, \ldots, 5C_h + 2\} \\
&\quad \cup \{5C_h + 3, \ldots, 6C_h + 1\} \text{ (only if } C_h \geqslant 2) .
\end{aligned}$$

Partition $\mathcal{S}_0^{(h)}$ represents those states with no servers in submodel h and 0 through C_h customers in the waiting space of queue h. Partition $\mathcal{S}_1^{(h)}$ represents those states with one server in submodel h. Note that this server can be traveling from queue h to the next queue in line or serving customers in queue h. We let states in $\{C_h + 1, \ldots, 2C_h + 1\}$ represent a traveling server with 0 through C_h customers in the waiting space and states in $\{2C_h+2, \ldots, 3C_h+1\}$

represent a serving server with 1 through C_h customers in the waiting space. Finally, partition $\mathcal{S}_2^{(h)}$ represents those states with two servers in submodel h. Of these two servers, both can be traveling, one can be traveling and the other serving, or both can be serving. Hence, we have the subpartitions $\{3C_h + 2, \ldots, 4C_h + 2\}$, $\{4C_h + 3, \ldots, 5C_h + 2\}$, and $\{5C_h + 3, \ldots, 6C_h + 1\}$, respectively. We remark that the last subpartition, $\{5C_h + 3, \ldots, 6C_h + 1\}$, exists only if $C_h \geqslant 2$. According to this understanding with $S = 2$ and letting $C_1 = C_H = 2$, in Figure 2.5 state of submodel 1 would be 7 and state of submodel H would be 3.

In general, there will be $S + 1$ partitions of $\mathcal{S}^{(h)}$ numbered 0 through S, where partition $\mathcal{S}_s^{(h)}$ represents those states in submodel h with s servers for $s = 0, \ldots, S$. Assuming that $C_h \geqslant S$, these s servers yield $(s + 1)$ subpartitions (from s traveling and 0 serving to 0 traveling and s serving) in $\mathcal{S}_s^{(h)}$, respectively, with sizes $C_h + 1$, C_h, ..., $C_h - s + 1$, hence altogether $|\mathcal{S}_s^{(h)}| = 1 + (s + 1)C_h - (s - 1)s/2$ states. This implies

$$|\mathcal{S}^{(h)}| = \sum_{s=0}^{S} |\mathcal{S}_s^{(h)}| = \frac{(S^2 + 3S + 2)C_h}{2} - \frac{(S^3 - 7S - 6)}{6} .$$

When $S = 2$ and $C_h \geqslant S$ for $h = 1, \ldots, H$, we have $|\mathcal{S}^{(h)}| = 6C_h + 2$.

Here, we provide a Kronecker representation of this model for two servers and three queues (i.e., $S = 2$, $H = 3$) with a capacity of two in each queue (i.e., $C_1 = C_2 = C_3 = 2$). Hence, for $h = 1, 2, 3$ we have

$$\mathcal{S}^{(h)} = \{0, \ldots, 13\} , \quad \text{thus, } |\mathcal{S}^{(h)}| = 14 ,$$

where

$$\mathcal{S}_0^{(h)} = \{0, 1, 2\}, \quad \mathcal{S}_1^{(h)} = \{3, 4, 5, 6, 7\}, \quad \text{and} \quad \mathcal{S}_2^{(h)} = \{8, 9, 10, 11, 12, 13\} .$$

We remark that $N = 6$ conveniently provides the number of partitions of the reachable state space, $\mathcal{R}$, to be used in this model. Partitions $\mathcal{R}_0$, $\mathcal{R}_1$, and $\mathcal{R}_2$ can be used for those states in which both servers are, respectively, at submodels 1, 2, and 3. Then partition $\mathcal{R}_3$ can represent one server at submodel 1, the other server at submodel 2. Partition $\mathcal{R}_4$ can represent one server at submodel 2, the other server at submodel 3, and partition $\mathcal{R}_5$ can represent one server at submodel 3, the other server at submodel 1.

With this understanding, the reachable state space can be expressed as

$$\mathcal{R} = \bigcup_{p=0}^{5} \mathcal{R}_p \quad \text{with} \quad \mathcal{R}_p = \bigtimes_{h=1}^{3} \mathcal{R}_p^{(h)} \quad \text{for } p = 0, \ldots, 5 ,$$

where

$$\mathcal{R}_0 = \mathcal{S}_2^{(1)} \times \mathcal{S}_0^{(2)} \times \mathcal{S}_0^{(3)} , \quad \mathcal{R}_1 = \mathcal{S}_0^{(1)} \times \mathcal{S}_2^{(2)} \times \mathcal{S}_0^{(3)} , \quad \mathcal{R}_2 = \mathcal{S}_0^{(1)} \times \mathcal{S}_0^{(2)} \times \mathcal{S}_2^{(3)} ,$$

$$\mathcal{R}_3 = \mathcal{S}_1^{(1)} \times \mathcal{S}_1^{(2)} \times \mathcal{S}_0^{(3)} , \quad \mathcal{R}_4 = \mathcal{S}_0^{(1)} \times \mathcal{S}_1^{(2)} \times \mathcal{S}_1^{(3)} , \quad \mathcal{R}_5 = \mathcal{S}_1^{(1)} \times \mathcal{S}_0^{(2)} \times \mathcal{S}_1^{(3)} .$$

The mapping between reachable state space partitions and submodel states is given in Table 2.2. Note that the size of the reachable state space is $|\mathcal{R}| = 387$, whereas that of the product state space is $|\mathcal{S}| = \prod_{h=1}^{3} |\mathcal{S}^{(h)}| = 14^3 = 2{,}744$.

Table 2.2 Mapping between reachable state space partitions and submodel states in polling model with three submodels

Partition	Submodel 1	Submodel 2	Submodel 3	# of states
$\mathcal{R}_0$	8:13	0:2	0:2	$6 \cdot 3 \cdot 3 = 54$
$\mathcal{R}_1$	0:2	8:13	0:2	$3 \cdot 6 \cdot 3 = 54$
$\mathcal{R}_2$	0:2	0:2	8:13	$3 \cdot 3 \cdot 6 = 54$
$\mathcal{R}_3$	3:7	3:7	0:2	$5 \cdot 5 \cdot 3 = 75$
$\mathcal{R}_4$	0:2	3:7	3:7	$3 \cdot 5 \cdot 5 = 75$
$\mathcal{R}_5$	3:7	0:2	3:7	$5 \cdot 3 \cdot 5 = 75$

In the particular model under consideration, $K = 6$ transitions numbered 1 through 6 take place among the states in the reachable state space partitions and affect the submodels. These transitions are captured by the (6×6) interaction matrix in (2.11) as

$$\begin{array}{c} \\ \mathcal{R}_0 \\ \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \\ \mathcal{R}_4 \\ \mathcal{R}_5 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathcal{R}_0 & \mathcal{R}_1 & \mathcal{R}_2 & \mathcal{R}_3 & \mathcal{R}_4 & \mathcal{R}_5 \end{array} \\ \begin{pmatrix} \{1,2,3\} & & & \{4\} & & \\ & \{1,2,3\} & & & \{5\} & \\ & & \{1,2,3\} & & & \{6\} \\ & \{4\} & & \{1,2,3\} & & \{5\} \\ & & \{5\} & \{6\} & \{1,2,3\} & \\ \{6\} & & & & \{4\} & \{1,2,3\} \end{pmatrix} \end{array} . \quad (2.12)$$

Transitions 1 through 3 represent the self evolution of submodels 1 through 3, respectively, upon customer arrivals and services. Given that there is space to accommodate a customer arrival in a submodel, the arrival causes the number of customers in the waiting space of its queue to increase by one, whereas a customer service completion causes the number of customers in the waiting space of its queue to decrease by one and the server to move to the traveling phase. As such, both of these events cause a state change only in the submodel they take place, and hence, they may be viewed as being local. On the other hand, transitions 4 through 6 model the travel of a server from submodel 1 to 2, from submodel 2 to 3, and from submodel 3 to 1, respectively. Therefore, each of the transitions 4 through 6 cause state changes in exactly two submodels, the submodel from which the server is departing and the submodel at which the server is arriving. In general, there are $K = 2H$ transitions in this model.

To each of the transitions 1 through 6 corresponds a Kronecker product of three submodel submatrices. The submatrices associated with those sub-

models that do not participate in a transition are all identity. Each submodel participates in three transitions. One of these transitions is local and represents customer arrivals and departures, the other two represent departures and arrivals of servers. Submodel 1 participates in transitions $1, 4, 6$, submodel 2 participates in transitions $2, 4, 5$, and submodel 3 participates in transitions $3, 5, 6$. For instance, submodel 1 participates in transition 1 with submatrix $Q_1^{(1)}(8:13, 8:13)$ when in $\mathcal{R}_0$, with submatrix $Q_1^{(1)}(0:2, 0:2)$ when in $\mathcal{R}_1$, $\mathcal{R}_2$, or $\mathcal{R}_4$, and with submatrix $Q_1^{(1)}(3:7, 3:7)$ when in $\mathcal{R}_3$ or $\mathcal{R}_5$. Submodel 1 participates in transition 4 with submatrix $Q_4^{(1)}(8:13, 3:7)$ when in $\mathcal{R}_0$ and with submatrix $Q_4^{(1)}(3:7, 0:2)$ when in $\mathcal{R}_3$ or $\mathcal{R}_5$. Finally, submodel 1 participates in transition 6 with submatrix $Q_6^{(1)}(0:2, 3:7)$ when in $\mathcal{R}_2$ or $\mathcal{R}_4$ and with submatrix $Q_6^{(1)}(3:7, 8:13)$ when in $\mathcal{R}_5$.

Transition h for $h = 1, 2, 3$ represents customer arrivals and services in submodel h with the matrices

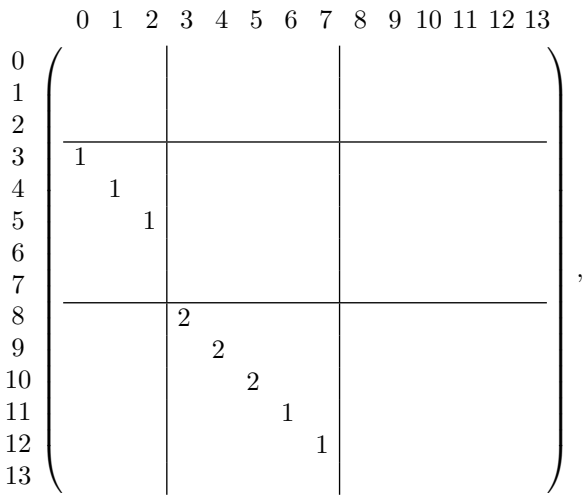
$$Q_h^{(h)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \end{array} \\ \left(\begin{array}{cc|cccccc|cccccc} \lambda_h & & & & & & & & & & & & & \\ & \lambda_h & & & & & & & & & & & & \\ \hline & & & \lambda_h & & & & & & & & & & \\ & & & & \lambda_h & & & & & & & & & \\ & & & & & & & & & & & & & \\ & & & \mu_h & & & & \lambda_h & & & & & & \\ & & & & \mu_h & & & & & & & & & \\ \hline & & & & & & & & \lambda_h & & & & & \\ & & & & & & & & & \lambda_h & & & & \\ & & & & & & & & & & & & & \\ & & & & & & & & \mu_h & & & & \lambda_h & \\ & & & & & & & & & \mu_h & & & & \\ & & & & & & & & & & & 2\mu_h & & \end{array}\right) \end{array} .$$

$$Q_1^{(2)} = Q_1^{(3)} = Q_2^{(1)} = Q_2^{(3)} = Q_3^{(1)} = Q_3^{(2)} = I_{14} .$$

Note that transitions due to arrivals appear in the upper-triangular part, whereas transitions due to services appear in the lower-triangular part of $Q_h^{(h)}$. The number of nonzeros in $Q_h^{(h)}$ due to arrivals is $(S^2 + 3S + 2)C_h/2 - (S^3 + 3S^2 + 2S)/6$ and due to services is $(S^2 + S)C_h/2 - (S^3 - S)/6$. Thus, $Q_h^{(h)}$ has a total of $(S^2 + 2S + 1)C_h - (2S^3 + 3S^2 + S)/6$ nonzeros when $C_h \geqslant S$ for $h = 1, \ldots, H$. When $H = 3$, $S = 2$, and $C_1 = C_2 = C_3 = 2$, this number is 13 in each of the H matrices corresponding to transitions 1 through H.

Transitions 4, 5, and 6 represent the travel of a server in the form of a departure from one submodel and an arrival to the next submodel in line with the matrices

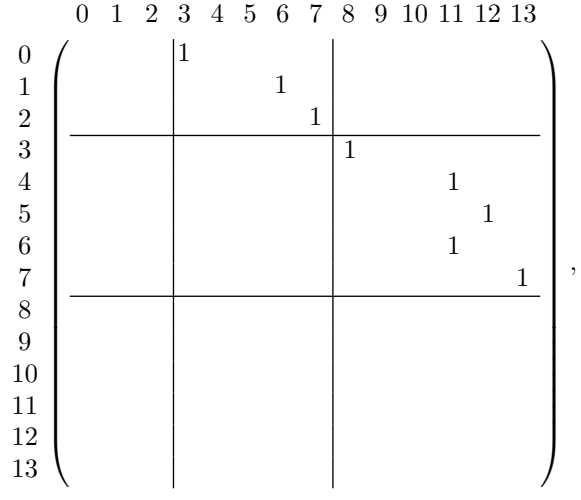
$$Q_4^{(1)} = Q_5^{(2)} = Q_6^{(3)} =$$



,

The number of nonzeros in the H submodel transition matrices due to server departures is $C_h(S^2+S)/2-(S^3-3S^2-4S)/6$. When $H=3$, $S=2$, and $C_1=C_2=C_3=2$, this number is 8 in each of the H matrices corresponding to transitions $H+1$ through $2H$.

$$Q_4^{(2)} = Q_5^{(3)} = Q_6^{(1)} =$$



,

$$Q_4^{(3)} = Q_5^{(1)} = Q_6^{(2)} = I_{14}\ .$$

The number of nonzeros in the H submodel transition matrices due to server arrivals is $C_h(S^2+S)/2-(S^3-3S^2-4S)/6$. When $H=3$, $S=2$, and $C_1=C_2=C_3=2$, this number is 8 in each matrix corresponding to transitions $H+1$ through $2H$.

The rates of transitions 1 through 6 are given by

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6) = (1, 1, 1, \gamma_1, \gamma_2, \gamma_3) \, .$$

As in (2.10), the polling model defines the (6×6) block generator matrix

$$Q = \begin{pmatrix} Q(0,0) & & & Q(0,3) & & \\ & Q(1,1) & & & Q(1,4) & \\ & & Q(2,2) & & & Q(2,5) \\ & Q(3,1) & & Q(3,3) & & Q(3,5) \\ & & Q(4,2) & Q(4,3) & Q(4,4) & \\ Q(5,0) & & & & Q(5,4) & Q(5,5) \end{pmatrix},$$

where the nonzero blocks are given from Table 2.2 and the interaction matrix in (2.12) by

$$\begin{aligned}
Q(0,0) &= \alpha_1 Q_1^{(1)}(8:13, 8:13) \otimes I_3 \otimes I_3 + \alpha_2 I_6 \otimes Q_2^{(2)}(0:2, 0:2) \otimes I_3 \\
&\quad + \alpha_3 I_6 \otimes I_3 \otimes Q_3^{(3)}(0:2, 0:2) + Q_D(0,0), \\
Q(0,3) &= \alpha_4 Q_4^{(1)}(8:13, 3:7) \otimes Q_4^{(2)}(0:2, 3:7) \otimes I_3, \\
Q(1,1) &= \alpha_1 Q_1^{(1)}(0:2, 0:2) \otimes I_6 \otimes I_3 + \alpha_2 I_3 \otimes Q_2^{(2)}(8:13, 8:13) \otimes I_3 \\
&\quad + \alpha_3 I_3 \otimes I_6 \otimes Q_3^{(3)}(0:2, 0:2) + Q_D(1,1), \\
Q(1,4) &= \alpha_5 I_3 \otimes Q_5^{(2)}(8:13, 3:7) \otimes Q_5^{(3)}(0:2, 3:7), \\
Q(2,2) &= \alpha_1 Q_1^{(1)}(0:2, 0:2) \otimes I_3 \otimes I_6 + \alpha_2 I_3 \otimes Q_2^{(2)}(0:2, 0:2) \otimes I_6 \\
&\quad + \alpha_3 I_3 \otimes I_3 \otimes Q_3^{(3)}(8:13, 8:13) + Q_D(2,2), \\
Q(2,5) &= \alpha_6 Q_6^{(1)}(0:2, 3:7) \otimes I_3 \otimes Q_6^{(3)}(8:13, 3:7), \\
Q(3,1) &= \alpha_4 Q_4^{(1)}(3:7, 0:2) \otimes Q_4^{(2)}(3:7, 8:13) \otimes I_3, \\
Q(3,3) &= \alpha_1 Q_1^{(1)}(3:7, 3:7) \otimes I_5 \otimes I_3 + \alpha_2 I_5 \otimes Q_2^{(2)}(3:7, 3:7) \otimes I_3 \\
&\quad + \alpha_3 I_5 \otimes I_5 \otimes Q_3^{(3)}(0:2, 0:2) + Q_D(3,3), \\
Q(3,5) &= \alpha_5 I_5 \otimes Q_5^{(2)}(3:7, 0:2) \otimes Q_5^{(3)}(0:2, 3:7), \\
Q(4,2) &= \alpha_5 I_3 \otimes Q_5^{(2)}(3:7, 0:2) \otimes Q_5^{(3)}(3:7, 8:13), \\
Q(4,3) &= \alpha_6 Q_6^{(1)}(0:2, 3:7) \otimes I_5 \otimes Q_6^{(3)}(3:7, 0:2), \\
Q(4,4) &= \alpha_1 Q_1^{(1)}(0:2, 0:2) \otimes I_5 \otimes I_5 + \alpha_2 I_3 \otimes Q_2^{(2)}(3:7, 3:7) \otimes I_5 \\
&\quad + \alpha_3 I_3 \otimes I_5 \otimes Q_3^{(3)}(3:7, 3:7) + Q_D(4,4),
\end{aligned}$$

$$
\begin{aligned}
Q(5,0) &= \alpha_6 Q_6^{(1)}(3:7,8:13) \otimes I_3 \otimes Q_6^{(3)}(3:7,0:2),\\
Q(5,4) &= \alpha_4 Q_4^{(1)}(3:7,0:2) \otimes Q_4^{(2)}(0:2,3:7) \otimes I_5,\\
Q(5,5) &= \alpha_1 Q_1^{(1)}(3:7,3:7) \otimes I_3 \otimes I_5 + \alpha_2 I_5 \otimes Q_2^{(2)}(0:2,0:2) \otimes I_5\\
&\quad + \alpha_3 I_5 \otimes I_3 \otimes Q_3^{(3)}(3:7,3:7) + Q_D(5,5).
\end{aligned}
$$

Each diagonal block $Q_D(p,p)$ for $p = 0,\ldots,5$ is defined as before and will not be given explicitly.

The number of floating-point values to be stored in this representation is $2H$ for the rates, $(S^2+2S+1)\sum_{h=1}^{H} C_h - H(2S^3+3S^2+S)/6$ for the values in the matrices representing the arrivals and services, and $(S^2+S)\sum_{h=1}^{H} C_h - H(S^3-3S^2-4S)/3$ for the server travel matrices, thus altogether,

$$(2S^2+3S+1)\sum_{h=1}^{H} C_h - H(4S^3-3S^2-7S-12)/6.$$

For $H = 3$, $S = 2$, and $C_1 = C_2 = C_3 = 2$, this number becomes 93, whereas the same number is 1,057 for $H = 7$, $S = 2$, and $C_h = 10$ for $h = 1,\ldots,H$.

On the other hand, the calculation of the number of nonzeros in the off-diagonal part of Q in this case is more complicated than that of the availability model. Therefore, let us assume that $S = 2$ in the following derivation. Regarding transition h for $h = 1,\ldots,H$ there is one reachable state space partition in which submodel h has two servers. The submatrix of $Q_h^{(h)}$ associated with this partition has $5C_h - 4$ nonzeros. The submatrices associated with all the other submodels which have zero servers are of order $C_l + 1$ identity matrices for $l \neq h$. Hence, the number of nonzeros due to this partition is given by

$$(5C_h - 4)\prod_{\substack{l=1\\ l\neq h}}^{H}(C_l+1).$$

Transition h appears in $H-1$ partitions in which submodel h has one server and one other submodel also has one server. The submatrix of $Q_h^{(h)}$ corresponding to these partitions has $3C_h - 1$ nonzeros, whereas one other submodel, say l_1, $l_1 \neq h$, has a submatrix that is of order $2C_{l_1}+1$ identity matrix and the remaining submodels all have submatrices that are of order $C_l + 1$ identity matrices for $l = 1,\ldots,H$ with $l \neq h, l \neq l_1$. Hence, the total number of nonzeros due to these $H-1$ partitions is given by

$$(3C_h - 1)\sum_{\substack{l_1=1\\ l_1\neq h}}^{H}(2C_{l_1}+1)\prod_{\substack{l=1\\ l\neq h, l\neq l_1}}^{H}(C_l+1).$$

Finally, transition h appears in $H(H-1)/2$ other reachable state partitions in which submodel h has zero servers and C_h nonzeros in its corresponding

submatrix of $Q_h^{(h)}$. In $H-1$ of these partitions, one other submodel has two servers, while the remaining submodels have zero servers. This other submodel, say l_1, $l_1 \neq h$, has a submatrix that is of order $3C_{l_1}$ identity matrix and the remaining submodels all have submatrices that are of order C_l+1 identity matrices for $l=1,\ldots,H$ with $l \neq h, l \neq l_1$. Hence, these $H-1$ partitions contribute a total of

$$C_h \sum_{\substack{l_1 \neq h \\ l_1=1}}^{H} 3C_{l_1} \prod_{\substack{l \neq h, l \neq l_1 \\ l=1}}^{H} (C_l+1)$$

nonzeros. In $(H-2)(H-1)/2$ of these partitions, two other submodels each have one server, while the remaining submodels have zero servers. Hence, these $(H-2)(H-1)/2$ partitions contribute a total of

$$C_h \sum_{l_1=1}^{H} \sum_{l_2=1}^{H} (2C_{l_1}+1)(2C_{l_2}+1) \prod_{\substack{l \neq h, l \neq l_1, l \neq l_2 \\ l=1}}^{H} (C_l+1)$$

nonzeros. Therefore, transitions 1 through H altogether contribute

$$\begin{aligned}
\sum_{h=1}^{H} \Bigg(& (5C_h-4) \prod_{\substack{l \neq h \\ l=1}}^{H} (C_l+1) \\
& + (3C_h-1) \sum_{\substack{l_1 \neq h \\ l_1=1}}^{H} (2C_{l_1}+1) \prod_{\substack{l \neq h, l \neq l_1 \\ l=1}}^{H} (C_l+1) \\
& + C_h \sum_{\substack{l_1 \neq h \\ l_1=1}}^{H} 3C_{l_1} \prod_{\substack{l \neq h, l \neq l_1 \\ l=1}}^{H} (C_l+1) \\
& + C_h \sum_{l_1=1}^{H} \sum_{l_2=1}^{H} (2C_{l_1}+1)(2C_{l_2}+1) \prod_{\substack{l \neq h, l \neq l_1, l \neq l_2 \\ l=1}}^{H} (C_l+1) \Bigg)
\end{aligned}$$

nonzeros to the off-diagonal part of Q.

Each of the transitions $H+1$ through $2H$ contribute to two reachable state space partitions in which a server can depart from a submodel, one corresponding to two servers and the other corresponding to one server in that submodel. In transition $H+h$, the departing server from submodel h will be entering submodel $1+h \bmod H$. For each of the partitions corresponding to two servers and one server in submodel h, in addition to the identity matrices of order C_l+1 that contribute to the Kronecker product, we have two rectangular submatrices from $Q_{H+h}^{(h)}$ and $Q_{H+h}^{(1+h \bmod H)}$ that also contribute to the Kronecker product. When submodel h has two servers,

the corresponding submatrices of $Q_{H+h}^{(h)}$ and $Q_{H+h}^{(1+h \bmod H)}$ have, respectively, $2C_h + 1$ and $C_{1+h \bmod H} + 1$ nonzeros. When submodel h has one server, submodel $1 + h \bmod H$ has either one server or zero servers. In the former case, the corresponding submatrices of $Q_{H+h}^{(h)}$ and $Q_{H+h}^{(1+h \bmod H)}$, respectively, have $C_h + 1$ and $2C_{1+h \bmod H} + 1$ nonzeros. In the latter case, the corresponding submatrix of $Q_{H+h}^{(h)}$ has $C_h + 1$ nonzeros, the submatrix associated with $Q_{H+h}^{(1+h \bmod H)}$ is an order $C_{1+h \bmod H} + 1$ identity matrix, and one of the other $H - 2$ submodels, say l_1, has 1 server, implying a corresponding submatrix of $2C_{l_1} + 1$ nonzeros. Therefore, transitions $H + 1$ through $2H$ altogether contribute

$$\sum_{h=1}^{H} \Bigg((2C_h + 1)(C_{1+h \bmod H} + 1) \prod_{\substack{l=1 \\ l \neq h, l \neq 1+h \bmod H}}^{H} (C_l + 1) + (C_h + 1)(2C_{1+h \bmod H} + 1) \prod_{\substack{l=1 \\ l \neq h, l \neq 1+h \bmod H}}^{H} (C_l + 1) + (C_h + 1)(C_{1+h \bmod H} + 1) \sum_{\substack{l_1=1 \\ l_1 \neq h, l_1 \neq 1+h \bmod H}}^{H} (2C_{l_1} + 1) \prod_{\substack{l=1 \\ l \neq h, l \neq 1+h \bmod H, l \neq l_1}}^{H} (C_l + 1) \Bigg)$$

nonzeros to the off-diagonal part of Q.

When $C_h = C$ for $h = 1, \ldots, H$ with $H \geqslant 3$, the contribution from transitions 1 through H becomes

$$H(C+1)^{H-3}\Big((5C-4)(C+1)^2 + (H-1)(3C-1)(2C+1)(C+1) + 3(H-1)C^2(C+1) + (H-2)(H-1)C(2C+1)^2/2 \Big),$$

and the contribution from transitions $H + 1$ through $2H$ becomes

$$H^2(C+1)^{H-1}(2C+1).$$

After summing up these, we obtain

$$H(C+1)^{H-3}\Big((C+1)(11HC^2 + 4HC - 4C^2 - 3) + (H-2)(H-1)C(2C+1)^2/2 \Big)$$

as the number of nonzeros in the off-diagonal part of Q when $S = 2$ and $C_h = C$ for $h = 1, \ldots, H$ with $H \geqslant 3$.

When $H = 3$, $S = 2$, and $C_1 = C_2 = C_3 = 2$, Q is of order 387 and the number of nonzeros in its off-diagonal part is obtained as 1,383, whereas when $H = 7$, $S = 2$, and $C_h = 10$ for $h = 1, \ldots, H$, Q is of order 1,863,521,121 and has a total of 15,321,499,039 nonzero off-diagonal entries.

Chapter 3
Avoiding Unreachable States

The previous chapter has provided various examples of multidimensional MCs that are used to model systems composed of a finite number of interacting submodels. Assuming that there are H such interacting submodels, the state space of submodel h was denoted by $\mathcal{S}^{(h)}$ with $\mathcal{S}^{(h)} \subseteq \mathbb{Z}_{\geqslant 0}$ for $h = 1, \ldots, H$, and $\mathcal{S} = \times_{h=1}^{H} \mathcal{S}^{(h)}$ represented the Cartesian product of the submodel state spaces. Because the Cartesian product of multiple submodel state spaces is used in its definition without any restriction on the state spaces themselves, $\mathcal{S}$ is called the H-dimensional product state space of the model.

In some models, each of the states in $\mathcal{S}$ can be occupied at a particular instant in time. However, in many models, this is not the case due to semantic constraints. For instance, consider a buffer of finite capacity B in a communications network which receives two types of packets. Although there can be a maximum of B packets of either type in the buffer at different times, their sum at a particular instant can never be exceeding B. Hence, if the tuple (i_1, i_2) is used to indicate buffer occupancy by the two types of packets, then in addition to $0 \leqslant i_h \leqslant B$ for $h = 1, 2$ the constraint $i_1 + i_2 \leqslant B$ must be true at all times. A consequence of this observation is that not all $|\mathcal{S}| = (B+1)^2$ states in $\mathcal{S} = \{0, \ldots, B\} \times \{0, \ldots, B\}$ are possible. In fact, only the $|\mathcal{R}| = (B+1)(B+2)/2$ states in $\mathcal{R} = \{(0,0), \ldots, (0,B), (1,0), \ldots, (1,B-1), \ldots, (B-1,0), (B-1,1), (B,0)\}$ can be occupied by the model. Set $\mathcal{R}$ is called the multidimensional *reachable state space* of the model since it differs from $\mathcal{S}$ by those unreachable states which the model can never occupy. Example 2 in Section 2.2 and the polling model in Section 2.3 are other examples in which this phenomenon (that is, $\mathcal{R}$ being a proper subset of $\mathcal{S}$) takes place.

Compact storage of the generator matrix underlying the multidimensional MC incident on $\mathcal{R}$ and efficient implementation of relevant analysis methods using Kronecker operations require $\mathcal{R}$ to be represented as a union of Cartesian products of subsets of submodel state spaces as discussed in Section 2.2.

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_3

In this chapter, two different solution approaches are reviewed for this problem which is identified as Cartesian product partitioning of multidimensional reachable state spaces [108, 111].

Let us start with a formal definition. Given $\mathcal{R} \subseteq \mathcal{S}$, set $\{\mathcal{R}_0, \ldots, \mathcal{R}_{N-1}\}$ is a *Cartesian product partitioning* of the H-dimensional reachable state space $\mathcal{R}$ with N partitions if

$$\mathcal{R}_p = \bigtimes_{h=1}^{H} \mathcal{R}_p^{(h)},$$

$$\mathcal{R}_p^{(h)} \subseteq \mathcal{S}^{(h)}, \quad \mathcal{R}_p^{(h)} \text{ is a set of consecutive integers} \quad \text{for } h = 1, \ldots, H,$$

$$\bigcup_{p=0}^{N-1} \mathcal{R}_p = \mathcal{R} \quad \text{and} \quad \mathcal{R}_p \cap \mathcal{R}_w = \varnothing \text{ for } p \neq w,$$

$p, w = 0, \ldots, N-1$, and $N \in \mathbb{Z}_{>0}$.

The aim of partitioning $\mathcal{R}$ as such is to eliminate unreachable states from the sets of rows and columns of the generator matrix so that unnecessary flops due to unreachable states can be avoided during analysis. Consequently, the generator matrix underlying the MC incident on $\mathcal{R}$ can be viewed as an $(N \times N)$ block matrix as in (2.10). The case $N = 1$ implies $\mathcal{R} = \mathcal{S}$, hence, no unreachable states. Now, let us recall some preliminary definitions from discrete computational geometry and graph theory so that we can comment on the state of the art regarding Cartesian product partitioning of reachable state spaces [108, 111].

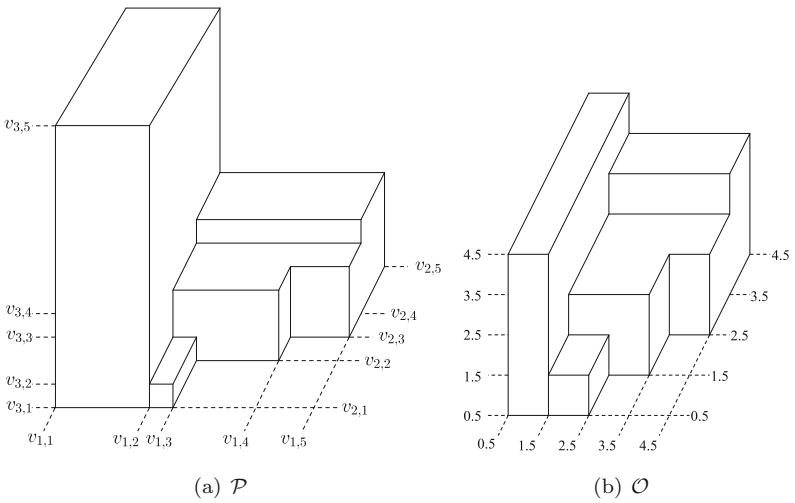
An H-dimensional *hyper-rectangle* is defined to be a Cartesian product of H intervals as in $\times_{h=1}^{H}[a_h, b_h]$, where $[a_h, b_h] \subseteq \mathbb{R}$ is an interval for $a_h < b_h$, $a_h, b_h \in \mathbb{R}$, and $h = 1, \ldots, H$. A point set $\mathcal{X} \subseteq \mathbb{R}^H$ is *convex* if the line segment between $\mathbf{x}$ and $\mathbf{y}$ is in $\mathcal{X}$ for $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. The *convex hull* of $\mathcal{X}$ is the smallest set containing $\mathcal{X}$. An H-dimensional *convex polytope* is the convex hull of a finite set $\mathcal{X} \subseteq \mathbb{R}^H$ [330]. An H-dimensional *polytope* is the union of a finite number of H-dimensional convex polytopes. An H-dimensional polytope is *orthogonal* if it is a union of H-dimensional hyper-rectangles [36].

Let $G = (\mathcal{V}, \mathcal{E})$ be an *undirected* graph with vertex (or node) set $\mathcal{V}$ and edge (or arc) set $\mathcal{E}$. A graph $G' = (\mathcal{V}', \mathcal{E}')$ is a *subgraph* of $G = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \{(\mathbf{i}, \mathbf{j}) \in \mathcal{E} \mid \mathbf{i}, \mathbf{j} \in \mathcal{V}'\}$. A subgraph of $G = (\mathcal{V}, \mathcal{E})$ induced by $\mathcal{V}' \subseteq \mathcal{V}$ is a graph whose vertex set is $\mathcal{V}'$ and edge set is $\{(\mathbf{i}, \mathbf{j}) \in \mathcal{E} \mid \mathbf{i}, \mathbf{j} \in \mathcal{V}'\}$. Two vertices of a graph are said to be connected if there exists a sequence of edges that lead from one of the vertices to the other. A subgraph of $G = (\mathcal{V}, \mathcal{E})$ forms a *connected component* if each pair of vertices in the subgraph is connected. A *unit distance graph* is a graph having a drawing in which all edges are of unit length [220].

It is known that an H-dimensional orthogonal polytope can be represented by a set of H-dimensional vectors [36]. In this representation, the Cartesian product of sets of consecutive integers corresponds to a hyper-rectangle. Hence, it can be concluded that Cartesian product partitioning of an H-dimensional reachable state space $\mathcal{R}$ is equivalent to the hyper-rectangular partitioning of the H-dimensional polytope that is represented by $\mathcal{R}$.

To aid the efficient use of Kronecker operations during analysis, the number of partitions, N, in the Cartesian product partitioning of $\mathcal{R}$ should be as small as possible. Interestingly, the partitioning of a two-dimensional orthogonal polytope into a minimum number of hyper-rectangles is a well-studied problem for which there are polynomial time algorithms [139, 193, 298]. Nevertheless, its three-dimensional version has been shown to be NP-complete [127], where NP stands for nondeterministic polynomial time. An algorithm to partition three-dimensional orthogonal polytopes into hyper-rectangles is proposed in [190]. To the best of our knowledge, [108, 111] provide the first algorithms to partition an H-dimensional orthogonal polytope into hyper-rectangles for $H > 3$.

In our context, the motivation is to automate the partitioning of a given multidimensional reachable state space $\mathcal{R}$ into Cartesian products of subsets of submodel state spaces. For practical purposes, the number of partitions, N, in the partitioning should be kept to a minimum. With this objective in mind, it is shown in [108] that the decision problem derived from the problem of partitioning $\mathcal{R}$ that corresponds to a three- or higher-dimensional model with a minimum number of partitions into Cartesian products of subsets of submodel state spaces is NP-complete [151].



(a) $\mathcal{P}$ (b) $\mathcal{O}$

Fig. 3.1 Three-dimensional orthogonal polytope $\mathcal{P}$ and its transformation $\mathcal{O}$

The proof is lengthy and starts with an H-dimensional orthogonal polytope $\mathcal{P}$, with $H \geqslant 3$, for which the problem is NP-complete. Then it introduces the transformation in Figure 3.1 to obtain another orthogonal polytope $\mathcal{O}$ which is a union of the same minimum number of hyper-rectangles as the original orthogonal polytope. Fortunately, this second orthogonal polytope $\mathcal{O}$ can be transformed to $\mathcal{R}$ which can be partitioned into a number of Cartesian

products of consecutive integers equal to the number of hyper-rectangles into which the polytope $\mathcal{O}$ can be partitioned. In doing this, points that are multiples of units along each dimension within polytope $\mathcal{O}$ are used. And this ends the proof whose details can be found in [108].

Being armed with an understanding of what is possible, in the next two sections, we will be presenting the two algorithms in [108, 111] that can be used to compute possibly non-optimal partitionings of $\mathcal{R}$ into Cartesian products of subsets of submodel state spaces. We assume without loss of generality that the submodel state spaces $\mathcal{S}^{(h)}$ are defined as before on consecutive nonnegative integers starting from 0 for $h = 1, \ldots, H$. Otherwise, it is always possible to enumerate $\mathcal{S}^{(h)}$ so that they satisfy this assumption. The first algorithm starts with partitions as singletons, each representing a reachable state. Two partitions are merged if their union is also a Cartesian product of sets of consecutive integers. The partitions are kept on being merged until there are no partitions that can be merged with each other. We call this the *merge-based* algorithm. The second algorithm takes a different approach. First, the unit distance graph of $\mathcal{R}$ is constructed. The vertex set of this graph is $\mathcal{R}$, and there is an edge between two vertices if the distance between them is one. Then this graph is refined [257] by removing edges until no further refinement is necessary. We call this the *refinement-based* algorithm. Through a set of problems from the literature including the availability and polling models in Section 2.3 and those that are randomly generated, the performance of the two algorithms will be investigated in this chapter. But, let us first introduce these algorithms in more detail.

3.1 Merge-Based Algorithm

The merge-based algorithm rests on the concept of *mergeability* of two partitions $\mathcal{I} = \times_{h=1}^{H} \mathcal{I}^{(h)}$ and $\mathcal{J} = \times_{h=1}^{H} \mathcal{J}^{(h)}$ in a Cartesian product partitioning of the H-dimensional reachable state space $\mathcal{R}$. Partitions $\mathcal{I}$ and $\mathcal{J}$ are said to be *mergeable* if $\mathcal{I} \cup \mathcal{J} = \times_{h=1}^{H} (\mathcal{I}^{(h)} \cup \mathcal{J}^{(h)})$ holds and $\mathcal{I}^{(h)} \cup \mathcal{J}^{(h)}$ consists of consecutive integers for $h = 1, \ldots, H$. Therefore, the mergeability condition can be stated formally as $\mathcal{I}$ and $\mathcal{J}$ are mergeable (along dimension l) if and only if there exists some $l \in \{1, \ldots, H\}$ such that

$$\max(\mathcal{I}^{(l)}) + 1 = \min(\mathcal{J}^{(l)}) \quad \text{or} \quad \max(\mathcal{J}^{(l)}) + 1 = \min(\mathcal{I}^{(l)}),$$

$$\text{and} \quad \mathcal{I}^{(h)} = \mathcal{J}^{(h)} \quad \text{for } h = 1, \ldots, l-1, l+1, \ldots, H.$$

The proof of this result is straightforward and can be found in [108]. Note that partitions $\mathcal{I}$ and $\mathcal{J}$ in their simplest forms can be singletons, and this is going to be the starting point for each state in $\mathcal{R}$ within the algorithm. The next example illustrates the concept of mergeability of partitions on a three-dimensional problem.

Example 3. Let $\mathcal{S}^{(h)} = \{0, 1, 2\}$ for $h = 1, 2, 3$ and $\mathcal{I} = \{2\} \times \{0, 1\} \times \{1\}$ be a partition in a Cartesian product partitioning of some $\mathcal{R}$ which satisfies $\mathcal{R} \subseteq \times_{h=1}^{3} \mathcal{S}^{(h)}$. Note that $\mathcal{I} = \{(2, 0, 1), (2, 1, 1)\}$ consists of two states, and due to the mergeability condition, it can only be merged with the Cartesian products $\{0, 1\} \times \{0, 1\} \times \{1\}$, $\{1\} \times \{0, 1\} \times \{1\}$, $\{2\} \times \{2\} \times \{1\}$, $\{2\} \times \{0, 1\} \times \{0\}$, and $\{2\} \times \{0, 1\} \times \{2\}$. For instance, merging $\mathcal{I}$ with $\{0, 1\} \times \{0, 1\} \times \{1\}$ results in the merged partition $\{0, 1, 2\} \times \{0, 1\} \times \{1\} = \{(0, 0, 1), (0, 1, 1), (1, 0, 1), (1, 1, 1), (2, 0, 1), (2, 1, 1)\}$, whereas merging $\mathcal{I}$ with $\{2\} \times \{2\} \times \{1\}$ yields the merged partition $\{2\} \times \{0, 1, 2\} \times \{1\} = \{(2, 0, 1), (2, 1, 1), (2, 2, 2)\}$.

Now, for partition $\mathcal{I} = \times_{h=1}^{H} \mathcal{I}^{(h)}$, the pair

$$(\min(\mathcal{I}^{(1)}), \ldots, \min(\mathcal{I}^{(H)})) \in \mathcal{I} \quad \text{and} \quad (\max(\mathcal{I}^{(1)}), \ldots, \max(\mathcal{I}^{(H)})) \in \mathcal{I}$$

defines its end states. Since partitions are mutually disjoint, so must be their end states. Hence, each partition can be specified by its two end states, and the states in a partition can be obtained from its end states. Consequently, the mergeability of two partitions can be determined by checking their pairs of end states only. Therefore, it is sufficient to specify partitions by storing only their end states instead of maintaining the partitions by using a disjoint-set data structure and a union-find algorithm [84] to merge two partitions.

In Algorithm 1, we provide the merge-based Cartesian product partitioning of $\mathcal{R}$. Note that this is a direct algorithm in which the body of the for-loop gets executed $|R|$ times and its efficiency depends on the data structure used to keep the end states of partitions. When a balanced binary search tree such as an AVL tree [1] is used to keep the end states (hence, two AVL trees, one for the low ends and one for the high ends in which each low end state points to its corresponding high end state in the other tree and vice versa are to be employed), the cost of searching for a mergeable partition becomes $O(H \lg(L))$ time. Here, L is the maximum number of partitions that exist during the execution of the algorithm, and $O(H)$ is the number of comparisons to be executed per compared partition since the end states in each dimension need to be checked for mergeability. Note that $1 \leqslant L \leqslant |\mathcal{R}|$, where $L = 1$ and $L = |\mathcal{R}|$ correspond to all states and no states merged cases in Algorithm 1, respectively.

At the outset, the Cartesian product partitioning of $\mathcal{R}$ denoted by $\mathcal{Q}$ is initialized to be the empty set. The algorithm constructs a singleton $\mathcal{I}$ for each state in $\mathcal{R}$ and seeks its mergeability with the already existing partitions in $\mathcal{Q}$. This requires comparing $\mathcal{I}$, the partition at hand, with each partition in $\mathcal{Q}$ to find a mergeable partition. If a mergeable partition $\mathcal{J}$ is located in $\mathcal{Q}$, $\mathcal{I}$ is merged with $\mathcal{J}$, $\mathcal{J}$ is deleted from $\mathcal{Q}$, and the algorithm continues as before, seeking a mergeable partition in $\mathcal{Q}$ with the larger partition $\mathcal{I}$ at hand until no mergeable partition is located. When the search ends for $\mathcal{I}$, it is inserted to $\mathcal{Q}$, and the algorithm continues with the next singleton coming from $\mathcal{R}$ as $\mathcal{I}$.

ALGORITHM 1. *Merge-based algorithm to compute a Cartesian product partitioning $\mathcal{Q}$ of given H-dimensional reachable state space $\mathcal{R}$.*

$\mathcal{Q} := \varnothing$;
For $\mathbf{i} \in \mathcal{R}$,
    $\mathcal{I} := \{\mathbf{i}\}$;
    While there exists some $\mathcal{J} \in \mathcal{Q}$ mergeable with $\mathcal{I}$,
        $\mathcal{I} := \mathcal{I} \cup \mathcal{J}$; $\mathcal{Q} := \mathcal{Q} \backslash \{\mathcal{J}\}$;
    $\mathcal{Q} := \mathcal{Q} \cup \{\mathcal{I}\}$.

Deletion and insertion operations each take $O(H \lg(L))$ time. Since the partitions in $\mathcal{Q}$ never get split, there can be at most $|\mathcal{R}| - 1$ merge operations in which each merge operation requires one deletion. Hence, the total number of deletions cannot be more than $|\mathcal{R}| - 1$. On the other hand, there is one insertion for each element in $\mathcal{R}$, implying $|\mathcal{R}|$ insertions altogether. Since the algorithm is executed for each state in $\mathcal{R}$, its time complexity becomes $O(H|\mathcal{R}| \lg(L))$, which is equal to $O(H|\mathcal{R}| \lg(|\mathcal{R}|))$ when no merge takes place. The space requirement for each partition is $O(H)$; hence, the space requirement of Algorithm 1 is $O(HL)$. When no partitions are merged, it becomes $O(H|\mathcal{R}|)$. If L is constant, the number of partitions is bounded by a constant during the execution of the algorithm. In that case, the time and space complexities of the algorithm become $O(H|\mathcal{R}|)$ and $O(H)$, respectively.

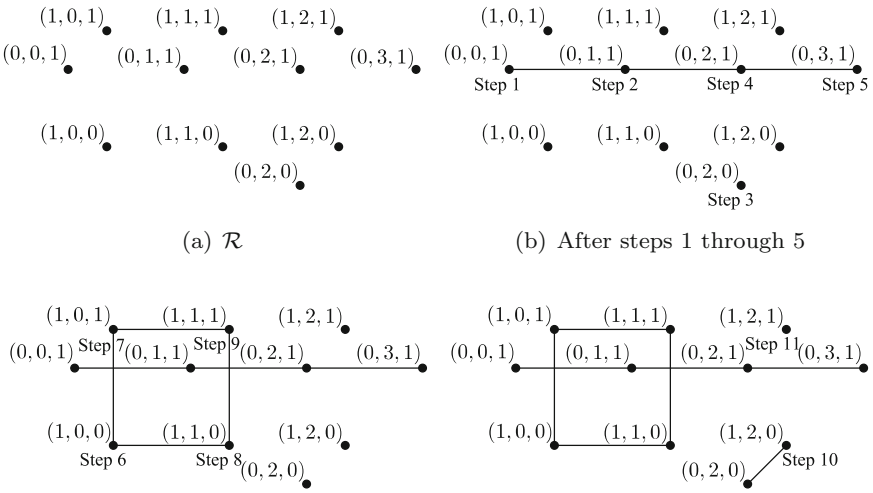
Observe that the partitioning $\mathcal{Q}$ computed by Algorithm 1 depends on the order in which the states of $\mathcal{R}$ are processed. Now, let us consider the following three-dimensional example.

Example 4. Let $\mathcal{S}^{(h)} = \{0, 1, 2, 3\}$ for $h = 1, 2, 3$,

$$\mathcal{R} = \{(0,0,1), (0,1,1), (0,2,0), (0,2,1), (0,3,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (1,2,0), (1,2,1)\},$$

and assume that the 11 states in $\mathcal{R}$ (see Figure 3.2(a)) are processed in lexicographical order.

Processing the first two states in $\mathcal{R}$ yields the partition $\{(0,0,1), (0,1,1)\}$. The partitions in $\mathcal{Q}$ become $\{(0,0,1), (0,1,1), (0,2,1)\}$ and $\{(0,2,0)\}$ after the next two states are processed. Then the singleton $\{(0,3,1)\}$ is merged with $\{(0,0,1), (0,1,1), (0,2,1)\}$ to give $\{(0,0,1), (0,1,1), (0,2,1), (0,3,1)\}$ and $\{(0,2,0)\}$ (see Figure 3.2(b)). The sixth and seventh states end up being merged together to yield the partitions $\{(0,0,1), (0,1,1), (0,2,1), (0,3,1)\}$, $\{(0,2,0)\}$, and $\{(1,0,0), (1,0,1)\}$. The eighth and ninth states are merged together to give $\{(1,1,0), (1,1,1)\}$, which in turn gets merged with $\{(1,0,0), (1,0,1)\}$ (see Figure 3.2(c)). Hence, before the last two states in $\mathcal{R}$ are processed, the partitions in $\mathcal{Q}$ have become



(a) $\mathcal{R}$ (b) After steps 1 through 5

(c) After steps 6 through 9 (d) After steps 10 and 11

Fig. 3.2 Steps of merge-based Cartesian product partitioning algorithm on $\mathcal{R}$ in Example 4

$$\{(0,0,1),(0,1,1),(0,2,1),(0,3,1)\}\ ,$$

$$\{(0,2,0)\} \quad \text{and} \quad \{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}\ .$$

The 10th state $(1,2,0)$ gets merged with the singleton $(0,2,0)$ to give $\{(0,2,0),(1,2,0)\}$, and the 11th state $(1,2,1)$ remains as a singleton (see Figure 3.2(d)).

The number of partitions in the Cartesian product partitioning of $\mathcal{R}$ processed in lexicographical order turns out to be four with Algorithm 1 as in

$$\mathcal{Q} = \{\{(0,0,1),(0,1,1),(0,2,1),(0,3,1)\},\{(0,2,0),(1,2,0)\},\\ \{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\},\{(1,2,1)\}\}.$$

3.2 Refinement-Based Algorithm

The refinement-based partitioning algorithm starts by constructing the unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of the H-dimensional reachable state space $\mathcal{R}$. The vertex set of the unit distance graph is $\mathcal{R}$, and its edge set is

$$\mathcal{E}^{(0)} = \{(\mathbf{i},\mathbf{j}) \mid \mathbf{i}-\mathbf{j} \in \bigcup_{h=1}^{H} \{-\mathbf{e}_h, \mathbf{e}_h\} \text{ for } \mathbf{i},\mathbf{j} \in \mathcal{R}\}\ .$$

In other words, two vertices in $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ are adjacent if there is consecutiveness of their state variables along a particular dimension while values of their state variables in other dimensions are the same. To construct $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$, the refinement-based algorithm requires processing all states in $\mathcal{R}$ before any refinement takes place, which implies a space requirement proportional to $|\mathcal{R}|$ at the very beginning. Note that this is not the case with the merge-based algorithm, which processes states one at a time. Let us now take a look at the unit distance graph of $\mathcal{R}$ in Example 4.

Example 4. (ctnd.) The unit distance graph of

$$\mathcal{R} = \{(0,0,1),(0,1,1),(0,2,0),(0,2,1),(0,3,1),(1,0,0),(1,0,1),(1,1,0),\\ (1,1,1),(1,2,0),(1,2,1)\}$$

with 11 vertices and 15 edges is depicted in Figure 3.3.

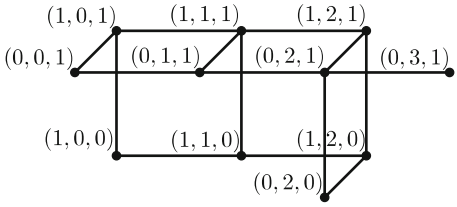


Fig. 3.3 Unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of $\mathcal{R}$ in Example 4

Now, we define *conflicting edges* in a subgraph $G = (\mathcal{R}, \mathcal{E})$ of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ as in [108, 111]. Let $\mathbf{i}, \mathbf{i} + \boldsymbol{\delta}_l, \mathbf{i} + \boldsymbol{\delta}_h \in \mathcal{R}$ for some $l, h \in \{1, \ldots, H\}$, $l \neq h$, $\boldsymbol{\delta}_l \in \{-\mathbf{e}_l, \mathbf{e}_l\}$, and $\boldsymbol{\delta}_h \in \{-\mathbf{e}_h, \mathbf{e}_h\}$. Two edges $(\mathbf{i}, \mathbf{i} + \boldsymbol{\delta}_l)$ and $(\mathbf{i}, \mathbf{i} + \boldsymbol{\delta}_h) \in \mathcal{E}$ are said to be conflicting if

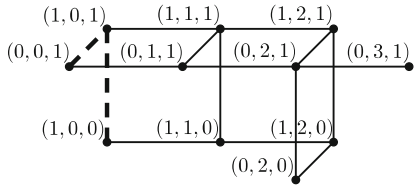
$$\mathbf{i} + \boldsymbol{\delta}_l + \boldsymbol{\delta}_h \notin \mathcal{R} \quad \text{or} \quad \{(\mathbf{i} + \boldsymbol{\delta}_l, \mathbf{i} + \boldsymbol{\delta}_l + \boldsymbol{\delta}_h), (\mathbf{i} + \boldsymbol{\delta}_h, \mathbf{i} + \boldsymbol{\delta}_l + \boldsymbol{\delta}_h)\} \nsubseteq \mathcal{E}.$$

We remark that the latter condition enables us to argue in terms of an arbitrary subgraph $G = (\mathcal{R}, \mathcal{E})$ of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ in the next results; otherwise, it will not be encountered in the devised algorithm. Let us now show the conflicting edges in the unit distance graph of $\mathcal{R}$ in Example 4.

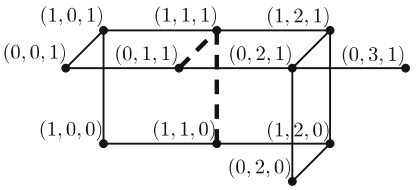
Example 4. (ctnd.) The unit distance graph of $\mathcal{R}$ has the six pairs of conflicting edges that are indicated by dashed line segments in Figure 3.4.

The next result and its corollary [108, 111] show that in a subgraph of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ without any conflicting edges, the vertices in each connected component can be written as a Cartesian product of sets of consecutive integers. Therefore, another way of obtaining a Cartesian product partitioning of $\mathcal{R}$ is to eliminate conflicting edges from $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$.

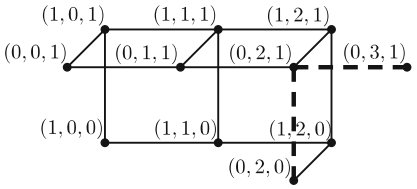
Consider a subgraph $G = (\mathcal{R}, \mathcal{E})$ of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ without any conflicting edges, $M \in \mathbb{Z}_{>0}$, $\mathbf{i}^{(m)} \in \mathcal{R}$ for $m = 0, \ldots, M$, and



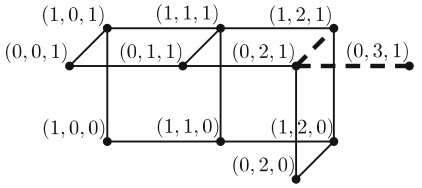
(a) ((0,0,1),(1,0,1)) and ((1,0,0),(1,0,1))



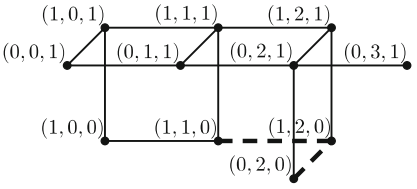
(b) ((0,1,1),(1,1,1)) and ((1,1,0),(1,1,1))



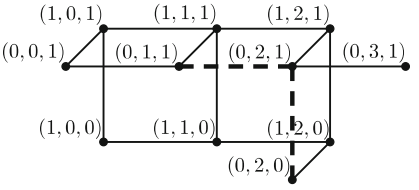
(c) ((0,2,0),(0,2,1)) and ((0,2,1),(0,3,1))



(d) ((0,2,1),(0,3,1)) and ((0,2,1),(1,2,1))



(e) ((0,2,0),(1,2,0)) and ((1,1,0),(1,2,0))



(f) ((0,1,1),(0,2,1)) and ((0,2,0),(0,2,1))

Fig. 3.4 Conflicting edges in the unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of $\mathcal{R}$ in Example 4

$$\mathcal{I}(M) = \bigtimes_{h=1}^{H} \{\min_{m=0}^{M}(i_h^{(m)}), \ldots, \max_{m=0}^{M}(i_h^{(m)})\}.$$

If $(\mathbf{i}^{(m-1)}, \mathbf{i}^{(m)}) \in \mathcal{E}$ for $m = 1, \ldots, M$, then $\mathcal{I}(M) \subseteq \mathcal{R}$ and $(\mathbf{i}, \mathbf{j}) \in \mathcal{E}$ for $\mathbf{i} - \mathbf{j} \in \bigcup_{h=1}^{H} \{-\mathbf{e}_h, \mathbf{e}_h\}$ and $\mathbf{i}, \mathbf{j} \in \mathcal{I}(M)$. The proof of this result which appears in [108] is by mathematical induction on m. Now, letting $G' = (\mathcal{R}', \mathcal{E}')$ be a connected component in $G = (\mathcal{R}, \mathcal{E})$, this result implies $\mathcal{R}'$ can be written as a Cartesian product of consecutive integers, that is, $\mathcal{R}' = \times_{h=1}^{H} \{i_h, \ldots, j_h\}$, where $\mathbf{i}, \mathbf{j} \in \mathcal{R}$ and $i_h \leqslant j_h$ for $h = 1, \ldots, H$.

The algorithm refines $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ and its subgraphs by using separators that are defined next [108, 111]. Let $\mathcal{I}, \mathcal{J} \subseteq \mathcal{R}$ and $G = (\mathcal{R}, \mathcal{E})$ be a subgraph of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$. The edge set $\mathcal{Z}$ with $\mathcal{Z} \subseteq \mathcal{E}$ is said to be a *separator* if it satisfies the four conditions:

i. each edge in $\mathcal{Z}$ is incident to a vertex in $\mathcal{I}$ and a vertex in $\mathcal{J}$,
ii. subgraphs of G induced by the sets $\mathcal{I}$ and $\mathcal{J}$ are each connected,
iii. there does not exist two vertices $\mathbf{i}, \mathbf{j} \in \mathcal{R}$ such that

$$(\mathbf{i}, \mathbf{i} + d\mathbf{e}_h) \in \mathcal{Z}, \quad (\mathbf{j}, \mathbf{j} + d\mathbf{e}_h) \in \mathcal{E} \backslash \mathcal{Z}, \quad \{(\mathbf{i}, \mathbf{j}), (\mathbf{i} + d\mathbf{e}_h, \mathbf{j} + d\mathbf{e}_h)\} \subseteq \mathcal{E}, \quad \text{and}$$

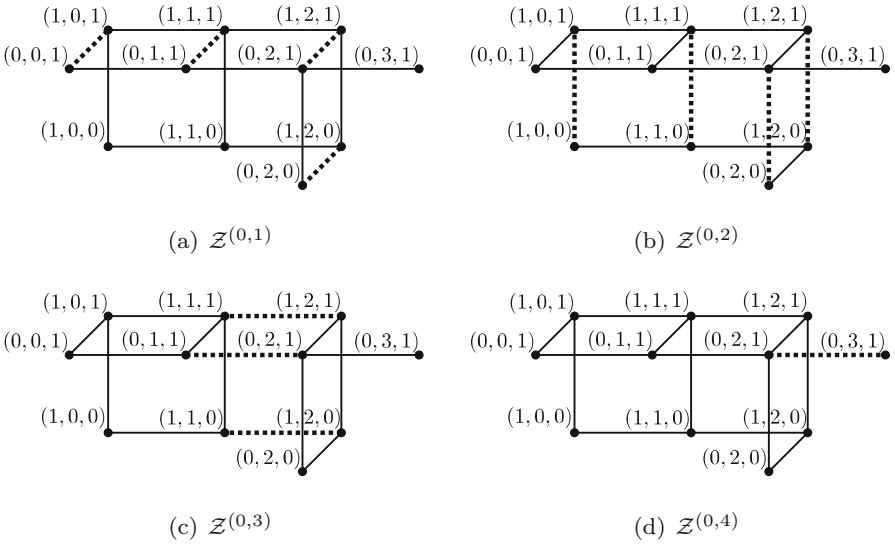
iv. there exists at least one edge in $\mathcal{Z}$ that conflicts with some edge in $\mathcal{E}$,

where $\mathcal{I} \subseteq \{\mathbf{i} \in \mathcal{R} \mid i_h = k\}$ and $\mathcal{J} \subseteq \{\mathbf{i} \in \mathcal{R} \mid i_h = k + d\}$ for some $k \in \mathcal{S}^{(h)}$, $d \in \{-1, 1\}$, and $h \in \{1, \ldots, H\}$. Let us now illustrate the separators in the unit distance graph of $\mathcal{R}$ in Example 4.

Example 4. (ctnd.) The unit distance graph of $\mathcal{R}$ has the four separators

$$\begin{aligned}
\mathcal{Z}^{(0,1)} &= \{((0,0,1),(1,0,1)),((0,1,1),(1,1,1)),((0,2,0),(1,2,0)),\\
&\quad ((0,2,1),(1,2,1))\}\,,\\
\mathcal{Z}^{(0,2)} &= \{((0,2,0),(0,2,1)),((1,0,0),(1,0,1)),((1,1,0),(1,1,1)),\\
&\quad ((1,2,0),(1,2,1))\}\,,\\
\mathcal{Z}^{(0,3)} &= \{((0,1,1),(0,2,1)),((1,1,0),(1,2,0)),((1,1,1),(1,2,1))\}\,,\\
\mathcal{Z}^{(0,4)} &= \{((0,2,1),(0,3,1))\}
\end{aligned}$$

that are indicated by dotted line segments in Figure 3.5.



(a) $\mathcal{Z}^{(0,1)}$ (b) $\mathcal{Z}^{(0,2)}$

(c) $\mathcal{Z}^{(0,3)}$ (d) $\mathcal{Z}^{(0,4)}$

Fig. 3.5 Separators in the unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of $\mathcal{R}$ in Example 4

To further explain how separators are formed, note that the two disjoint sets of $\mathcal{Z}^{(0,1)}$ are given by

$$\begin{aligned}\mathcal{I}^{(0,1)} &= \{(0,0,1),(0,1,1),(0,2,0),(0,2,1)\}\\ \mathcal{J}^{(0,1)} &= \{(1,0,1),(1,1,1),(1,2,0),(1,2,1)\}\ ;\end{aligned}$$

the subgraphs of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ induced by $\mathcal{I}^{(0,1)}$ and $\mathcal{J}^{(0,1)}$ are each connected as in Figure 3.6; and the separator consists of edges each of which has constant values in its incident multidimensional vertices except dimension 1. A similar line of reasoning applies to each of the other three separators.

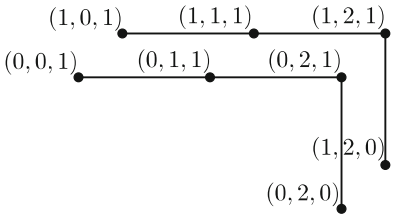


Fig. 3.6 Subgraphs induced by $\mathcal{I}^{(0,1)}$ and $\mathcal{J}^{(0,1)}$ corresponding to separator $\mathcal{Z}^{(0,1)}$ in unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of $\mathcal{R}$ for Example 4

Now, let $\mathcal{Z}$ be a separator in $G = (\mathcal{R}, \mathcal{E})$, a subgraph of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$. Two edges in $G' = (\mathcal{R}, \mathcal{E} \backslash \mathcal{Z})$ conflict only if they also conflict in $G = (\mathcal{R}, \mathcal{E})$ [108, 111]. The proof of this result in [108] follows from the definitions of conflicting edge and separator. Hence, removing the edges in a separator decreases the number of conflicting edges in a subgraph of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$. Having provided all the ingredients, we are in a position to formally present the refinement-based Cartesian product partitioning of $\mathcal{R}$ in Algorithm 2.

First, the unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of $\mathcal{R}$ is constructed. This requires identifying all adjacent vertices in $\mathcal{R}$, which can be done by letting each vertex in $\mathcal{R}$ keep an adjacency list of length $2H$ since there are H dimensions and adjacent vertices in each dimension can be in the plus or minus directions. To facilitate this, vertices in $\mathcal{R}$ are first inserted to an AVL tree [1], and then for each vertex, vertices that might be adjacent to it in the product state space $\mathcal{S}$ are sought in the tree. Since there are $2H$ such possibilities for each vertex, $2H$ vertices end up being sought for each vertex. Therefore, maintaining the graph requires a space complexity of $O(H|\mathcal{R}|)$, and constructing $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ in this way suggests a time complexity of $O(H|\mathcal{R}| \lg(|\mathcal{R}|))$. The AVL tree is destroyed after the graph is constructed.

ALGORITHM 2. *Refinement-based algorithm to compute a Cartesian product partitioning $\mathcal{Q}$ of given H-dimensional reachable state space $\mathcal{R}$.*

Construct unit distance graph $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ of $\mathcal{R}$;
Construct an empty priority queue $\mathcal{PQ}$;
For $\mathbf{i} \in \mathcal{R}$,
  For $h := 1, \ldots, H$,
    If $\mathbf{i} + \mathbf{e}_h \in \mathcal{R}$,
      If $(\mathbf{i}, \mathbf{i} + \mathbf{e}_h)$ conflicts with some edge in $\mathcal{E}^{(0)}$
        and $(\mathbf{i}, \mathbf{i} + \mathbf{e}_h)$ is not in a separator,
          Construct separator $\mathcal{Z}$ including $(\mathbf{i}, \mathbf{i} + \mathbf{e}_h)$;
          Insert $\mathcal{Z}$ to $\mathcal{PQ}$;
$\mathcal{E} := \mathcal{E}^{(0)}$;
While $\mathcal{PQ}$ is not empty,
  $\mathcal{Z}_{\max} :=$ separator $\mathcal{Z}$ in $\mathcal{PQ}$ whose priority is maximum;
  Remove $\mathcal{Z}_{\max}$ from $\mathcal{PQ}$;
  For $(\mathbf{i}, \mathbf{j}) \in \mathcal{Z}_{\max}$,
    $\mathcal{L} := \varnothing$;
    For separators $\mathcal{Z}$ in $\mathcal{PQ}$ including an edge incident to $\mathbf{i}$ or $\mathbf{j}$,
      $\mathcal{L} := \mathcal{L} \cup \{\mathcal{Z}\}$; Remove $\mathcal{Z}$ from $\mathcal{PQ}$;
  $\mathcal{E} := \mathcal{E} \backslash \mathcal{Z}_{\max}$;
  For $\mathcal{Z} \in \mathcal{L}$,
    While $\mathcal{Z} \neq \varnothing$,
      Construct the separator $\mathcal{Z}'$ including some edge $(\mathbf{i}, \mathbf{j}) \in \mathcal{Z}$;
      $\mathcal{Z} := \mathcal{Z} \backslash \mathcal{Z}'$;
      If priority of $\mathcal{Z}'$ is positive,
        Insert $\mathcal{Z}'$ to $\mathcal{PQ}$;
$\mathcal{Q} := \varnothing$;
For connected components $G' = (\mathcal{R}', \mathcal{E}')$ of $G = (\mathcal{R}, \mathcal{E})$,
  $\mathcal{Q} := Q \cup \{\mathcal{R}'\}$.

Once $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ is constructed, its separators are formed and inserted to a priority queue [84] as in the for-loop on $\mathbf{i}$ in Algorithm 2, where the *priority of a separator* is defined as the total number of edges that conflict with the edges in the separator. Since each separator includes at least one conflicting edge due to the fourth item in the definition of a separator, priorities of separators are necessarily positive. Now, let us again turn to Example 4 and determine the priorities of the separators in $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$.

Example 4. (ctnd.) Recall that there are four separators in the unit distance graph of $\mathcal{R}$ as depicted in Figure 3.5. Of these separators, the edges in $\mathcal{Z}^{(0,1)}$ of Figure 3.5(a), respectively, conflict with the edges $((1,0,0),(1,0,1))$ in Figure 3.4(a), $((1,1,0),(1,1,1))$ in Figure 3.4(b), $((1,1,0),(1,2,0))$ in Figure 3.4(e), and $((0,2,1),(0,3,1))$ in Figure 3.4(d). Therefore, the priority of $\mathcal{Z}^{(0,1)}$ is 4. On the other hand, the first edge in $\mathcal{Z}^{(0,2)}$ of Figure 3.5(b) conflicts with the edges $((0,1,1),(0,2,1))$ in Figure 3.4(f) and

$((0,2,1),(0,3,1))$ in Figure 3.4(c); the second and third edges in $\mathcal{Z}^{(0,2)}$, respectively, conflict with the edges $((0,0,1),(1,0,1))$ in Figure 3.4(a) and $((0,1,1),(1,1,1))$ in Figure 3.4(b). Hence, the priority of $\mathcal{Z}^{(0,2)}$ is also 4. The first and second edges in $\mathcal{Z}^{(0,3)}$ of Figure 3.5(c), respectively, conflict with the edges $((0,2,0),(0,2,1))$ in Figure 3.4(f) and $((0,2,0),(1,2,0))$ in Figure 3.4(e), implying a priority of 2 for $\mathcal{Z}^{(0,3)}$. Finally, the only edge in $\mathcal{Z}^{(0,4)}$ of Figure 3.5(d) conflicts with the edges $((0,2,0),(0,2,1))$ in Figure 3.4(c) and $((0,2,1),(1,2,1))$ in Figure 3.4(d), implying also a priority of 2 for $\mathcal{Z}^{(0,4)}$.

Algorithm 2 constructs separators of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ by visiting each edge in $\mathcal{E}^{(0)}$ and checking whether it conflicts with some other edge. Recall that a separator is formed by two vertex sets $\mathcal{I}$ and $\mathcal{J}$ as in the first item in the definition of a separator with connected subgraphs induced by each of them as in the second item in the definition of a separator. Besides, the separator including a particular edge is unique due to the third item in the definition of a separator. Therefore, the separator including an edge is constructed only once for $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$. While constructing separator $\mathcal{Z}$ including the edge $(\mathbf{i}, \mathbf{j})$, a breadth-first search starting at $\mathbf{i}$ is used to visit the vertices connected to $\mathbf{i}$ [84]. Hence, the time complexity of constructing separator $\mathcal{Z}$ is $O(H|\mathcal{Z}|)$. Since each edge is added to at most one separator, the number of edges in the union of separators needs to be $O(|\mathcal{E}^{(0)}|)$. Therefore, the total time complexity of constructing all separators in $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ becomes $O(H|\mathcal{E}^{(0)}|)$. At each vertex, an array of size $2H$ is used to keep the separators including the edge incident to that vertex. The space complexity of the algorithm remains at $O(H|\mathcal{R}|)$, yet it takes constant time to retrieve the separator including a given edge.

When a separator is constructed, it is added to a priority queue that is implemented as a binary heap since later when refining the unit distance graph we need to access the separator with maximum priority rapidly at each refinement step. We remark that the number of separators cannot exceed $|\mathcal{E}^{(0)}|$ because each separator includes at least one edge. Therefore, the cost of inserting separators of $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ to the priority queue is $O(|\mathcal{E}^{(0)}| \lg(|\mathcal{E}^{(0)}|))$. An array implementation of the priority queue is used. Since the maximum number of separators is not known in advance, $O(|\mathcal{E}^{(0)}|) = O(H|\mathcal{R}|)$ space is allocated for the priority queue.

The graph is refined by removing the separator with maximum priority at each refinement step until no conflicting edges remain as in the while-loop terminating on an empty priority queue in Algorithm 2. Obviously, the edge set of the graph changes when a separator is removed during a refinement step; hence, the separators need to be reconstructed after a separator removal. We have already argued that an edge will be conflicting in the refined graph only if it is also conflicting before the refinement step. Therefore, each separator of the refined graph turns out to be a subset of a separator in the graph before refinement. This implies that in order to reconstruct separators, it is necessary and sufficient to visit vertices incident to edges in the separators intersecting the removed separator, where two separators are said to be *intersecting* if they include edges along different dimensions that are incident

to the same vertex. When all conflicting edges are eliminated, vertices in each connected component can be written as a union of Cartesian products of sets of consecutive integers in the last statement of Algorithm 2 using a for-loop as discussed before. Now, let us determine the number of intersecting separators in $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$.

Example 4. (ctnd.) The number of intersecting separators in the unit distance graph of $\mathcal{R}$ is five. Each pair of the four separators $\mathcal{Z}^{(0,1)}$, $\mathcal{Z}^{(0,2)}$, $\mathcal{Z}^{(0,3)}$, and $\mathcal{Z}^{(0,4)}$ is intersecting except the pair $\mathcal{Z}^{(0,3)}$ and $\mathcal{Z}^{(0,4)}$. Although these two pairs, respectively, have the edges $((0,1,1),(0,2,1))$ and $((0,2,1),(0,3,1))$ incident to the same vertex $(0,2,1)$, the two edges are along the same dimension, that is, dimension 2, and therefore not intersecting.

Now, let us explain the refinement process in some more detail. After inserting the separators in $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$ to the priority queue and setting $\mathcal{E} := \mathcal{E}^{(0)}$, at each refinement step, $G = (\mathcal{R}, \mathcal{E})$ is refined by removing the edges in its separator with maximum priority, $\mathcal{Z}_{\max}$. Note that the unit distance graph can be refined at most $|\mathcal{E}|$ times, because at each refinement step at least one of its edges is to be removed. Upon removing $\mathcal{Z}_{\max}$, its intersecting separators need to be reconstructed, meaning all edges of $\mathcal{Z}_{\max}$ are visited to obtain the intersecting separators so that they can be inserted to a list $\mathcal{L}$ and removed from the priority queue. Then these separators are reconstructed for the edge set $\mathcal{E} \backslash \mathcal{Z}_{\max}$ and reinserted to the priority queue. Reconstruction of a separator $\mathcal{Z}$ requires visiting all vertices incident to its edges. The number of edges in the union of separators intersecting with $\mathcal{Z}_{\max}$ is $O(H|\mathcal{E}|)$, which implies a time complexity of $O(H|\mathcal{E}|)$ at each refinement step for the reconstruction of separators. This yields a total time complexity of $O(H|\mathcal{E}|^2)$ for reconstructing separators since reconstruction is required in each refinement step. Now, let us turn to costs associated with priority queue operations. At each refinement step, separators intersecting with $\mathcal{Z}_{\max}$ need to be removed from the priority queue. Separator $\mathcal{Z}_{\max}$ intersects with $O(H|\mathcal{Z}_{\max}|)$ separators. Therefore, during execution of the algorithm, the total number of removals from the priority queue is $O(H|\mathcal{E}|)$ since the total size of separators removed from the graph $G = (\mathcal{R}, \mathcal{E})$ is $O(|\mathcal{E}|)$. The algorithm starts and ends with an empty priority queue, suggesting also $O(H|\mathcal{E}|)$ as the number of insertions to the priority queue. Hence, the total time complexity of priority queue operations is $O(H|\mathcal{E}| \lg(H|\mathcal{E}|))$. When all costs are considered, time and space complexities of Algorithm 2 are $O(H|\mathcal{E}^{(0)}|^2) = O(H^3|\mathcal{R}|^2)$ and $O(H|\mathcal{R}|)$, respectively.

Example 4. (cntd.) Let $G^{(k)} = (\mathcal{R}, \mathcal{E}^{(k)})$ denote the subgraph of the unit distance graph of $\mathcal{R}$ with edge set $\mathcal{E}^{(k)}$ after k separators are removed. There are two separators with maximum priority in $G^{(0)} = (\mathcal{R}, \mathcal{E}^{(0)})$, and one of them needs to be chosen. Let the edges in $\mathcal{Z}^{(0,1)}$ be chosen for removal. Then all other separators are reconstructed since the number of intersecting separators with $\mathcal{Z}^{(0,1)}$ is three and two pairs of conflicting edges in Figure 3.4(c) and Figure 3.4(f) remain in $G^{(1)} = (\mathcal{R}, \mathcal{E}^{(1)})$ as shown in Figure 3.7.

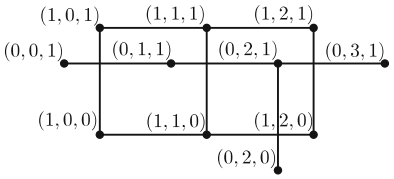


Fig. 3.7 Subgraph $G^{(1)} = (\mathcal{R}, \mathcal{E}^{(1)})$ in Example 4

Subgraph $G^{(1)} = (\mathcal{R}, \mathcal{E}^{(1)})$ has the three separators

$$\mathcal{Z}^{(1,1)} = \{((0,2,0),(0,2,1))\}\ ,\ \mathcal{Z}^{(1,2)} = \{((0,1,1),(0,2,1))\}\ ,\ \mathcal{Z}^{(1,3)} = \mathcal{Z}^{(0,4)}$$

that are indicated by dotted line segments in Figure 3.8. The priority of separator $\mathcal{Z}^{(1,1)}$ is 2 since its only edge conflicts with the edges $((0,2,1),(0,3,1))$ in Figure 3.4(c) and $((0,1,1),(0,2,1))$ in Figure 3.4(f). The priorities of the other two separators are 1 since the only edge in $\mathcal{Z}^{(1,2)}$ conflicts with the

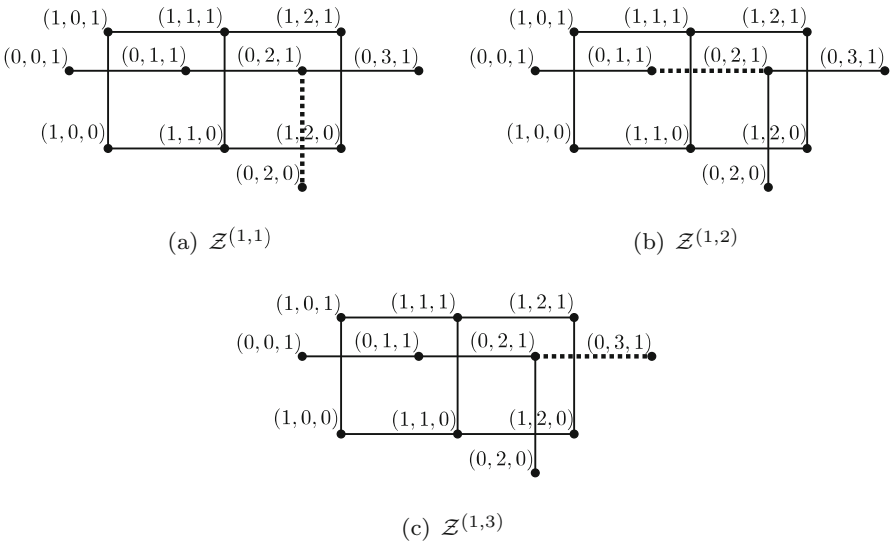


Fig. 3.8 Separators in subgraph $G^{(1)} = (\mathcal{R}, \mathcal{E}^{(1)})$ in Example 4

edge $((0,2,0),(0,2,1))$ in Figure 3.4(f) and the only edge in $\mathcal{Z}^{(1,3)}$ conflicts with the edge $((0,2,0),(0,2,1))$ in Figure 3.4(c). Hence, separator $\mathcal{Z}^{(1,1)}$ is chosen for removal from the edge set $\mathcal{E}^{(1)}$. After the edges in the separator are removed, $\mathcal{Z}^{(1,2)}$ and $\mathcal{Z}^{(1,3)}$ are reconstructed since both separators are intersecting with $\mathcal{Z}^{(1,1)}$ and no conflicting edges remain in $G^{(2)} = (\mathcal{R}, \mathcal{E}^{(2)})$ as in Figure 3.9.

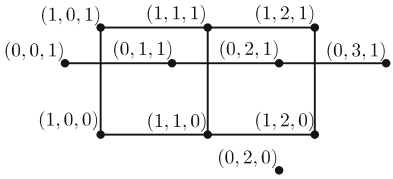


Fig. 3.9 Subgraph $G^{(2)} = (\mathcal{R}, \mathcal{E}^{(2)})$ in Example 4

Since subgraph $G^{(2)} = (\mathcal{R}, \mathcal{E}^{(2)})$ does not possess any conflicting edges, the vertices in each of its connected components are members of the same partition of the Cartesian product partitioning. Therefore, the number of partitions in the Cartesian product partitioning of $\mathcal{R}$ with Algorithm 2 is three and given by

$$\mathcal{Q} = \{\{(0,0,1),(0,1,1),(0,2,1),(0,3,1)\},\{(0,2,0)\}\ ,\\ \{(1,0,0),(1,0,1),(1,1,0),(1,1,1),(1,2,0),(1,2,1)\}\}\ .$$

Note that the number of partitions obtained with the refinement-based algorithm is smaller than that obtained with the merge-based algorithm.

3.3 Two Other Kronecker-Structured Models

In this section, we introduce two other Kronecker-structured models from the literature that will also serve as benchmarks. The production line model with Kanban control [245] we discuss next has a reachable state space that is equal to its product state space. The communications model using a Courier protocol between adjacent network layers [328] we investigate in the following subsection has unreachable states in its product state space. We think it is crucial to see how the developed algorithms perform on these examples as well.

3.3.1 A Production Line Model

We consider a large-scale production line in the form of a tandem of H submodels as in Figure 3.10, where each submodel represents a production unit. Each submodel (except the first and last in the tandem) consists of a machine that processes parts and its buffer, an output hopper where processed parts wait to be admitted to the next submodel in line, and a fixed number of cards allocated to the submodel. The control and coordination of parts that flow through each submodel to a finished product at the end of the line is handled

by the fixed number of cards, C_h, allocated to submodel h for $h = 1, \ldots, H$. Such a scheduling discipline is named Kanban due to the Japanese term used for the card and has been employed originally in the automotive industry to increase efficiency and productivity.

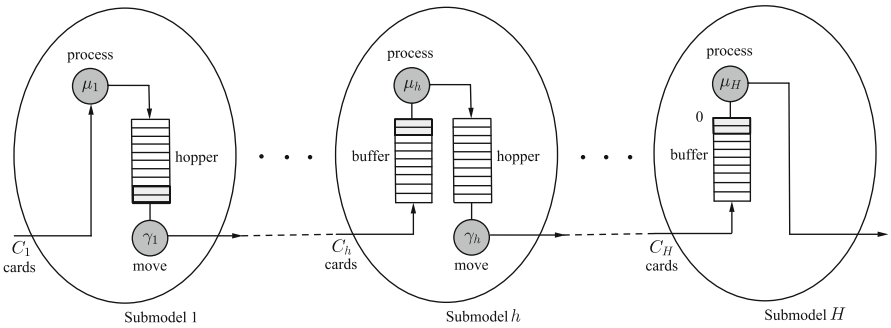


Fig. 3.10 Production line model with H submodels

Admission of parts to submodels is controlled strictly by the cards. If there are any unused cards in submodel h and a processed part in submodel $h-1$ is waiting to be admitted to submodel h, one of the unused cards in submodel h is marked as being used, and the part is admitted to the submodel. An arriving part at submodel h joins the buffer of the machine which processes incoming parts on a first-come first-served basis in an exponentially distributed amount of time with rate μ_h. Once processing of a part by the machine in submodel h finishes, the part joins the output hopper in which it starts waiting to be moved to submodel $h+1$. Moving of a processed part in submodel h to the next submodel in line takes an exponentially distributed amount of time with rate γ_h and is only possible if there is at least one unused card in submodel $h+1$. Otherwise, the processed part keeps on waiting in the output hopper until a card becomes available in submodel $h+1$. Upon moving to the next submodel, the card used by the processed part in submodel h is unmarked and becomes available for reuse. Submodels 1 and H operate slightly different than intermediate submodels; submodel 1 is assumed to always start processing a new part when a processed part moves from submodel 1 to submodel 2 implying there is no need for a machine buffer in submodel 1, and a processed part in submodel H immediately leaves the model implying there is no need for an output hopper in submodel H.

Assuming that i_h denotes the number of used cards in submodel h, Kanban control dictates that submodel h must have exactly i_h parts either associated with the machine or the hopper, where $0 \leqslant i_h \leqslant C_h$. In other words, the number of parts associated with the machine and the number of processed parts associated with the hopper in submodel h must sum up to i_h. With this understanding, submodels 1 and H can be represented using $C_1 + 1$ and $C_H + 1$ states, respectively, whereas submodel h needs to be represented with

$(C_h+1)(C_h+2)/2$ states for $h = 2, \ldots, H-1$ due to the number of different ways in which the i_h parts can be distributed between the machine buffer and the output hopper for $i_h = 0, \ldots, C_h$ (see also [50]). Note that had we not chosen to model the intermediate production unit h as a single submodel and therefore avoided grouping the machine and the hopper together, we would have unreachable states in its product state space of size $(C_h+1)^2$. Nevertheless, it is still necessary to allocate machine buffer and output hopper spaces each of size C_h in submodel h. The concept of grouping will be discussed in more detail in Chapter 4.

Without loss of generality, we assume that the $(C_h+1)(C_h+2)/2$ states of submodel h are ordered according to an increasing number of parts associated with the machine and according to an increasing number of processed parts associated with the hopper within each subset of states for a particular value of the number of parts associated with the machine. In other words, the first $C_h + 1$ states numbered 0 through C_h correspond to those in which there are no parts associated with the machine and 0 through C_h processed parts associated with the hopper. The next C_h states numbered $C_h + 1$ through $2C_h$ correspond to those states in which there is one part associated with the machine and 0 through $C_h - 1$ processed parts associated with the hopper, and so on. The last state numbered $(C_h + 1)(C_h + 2)/2 - 1$ corresponds to C_h processed parts associated with the hopper.

The submodel state spaces are given by

$$\mathcal{S}^{(1)} = \{0, \ldots, C_1\}\,, \quad \mathcal{S}^{(H)} = \{0, \ldots, C_H\}\,, \quad \text{and}$$

$$\mathcal{S}^{(h)} = \{0, \ldots, (C_h + 1)(C_h + 2)/2 - 1\} \text{ for } h = 2, \ldots, H-1.$$

Hence, for this H-dimensional production line model, we have a reachable state space equal to its product state space containing

$$|\mathcal{S}| = (C_1 + 1)(C_H + 1) \prod_{h=2}^{H-1} (C_h + 1)(C_h + 2)/2$$

states. We will be considering models with $H = 4, 5, 6, 7, 8$.

There are altogether $K = 2H - 1$ transitions numbered 1 through $2H - 1$ in the model, the first H of which may be perceived as being local. On the other hand, transitions $H + 1$ to $2H - 1$ represent synchronizations between adjacent submodels in such a way that transition $H + h$ models the departure of a processed part from submodel h and its arrival to submodel $h+1$ thereby involving two adjacent submodels for $h = 1, \ldots, H-1$. Here, we provide a Kronecker representation of this model for $C_h = 4$ and $h = 1, \ldots, H$.

Transition 1 represents the self-evolution of submodel 1 through which parts enter the model, with the matrices

$$Q_1^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} & \mu_1 & & & \\ & & \mu_1 & & \\ & & & \mu_1 & \\ & & & & \mu_1 \\ & & & & \end{pmatrix} \end{array},$$

$$Q_1^{(H)} = I_{C_H+1}, \quad \text{and} \quad Q_1^{(h)} = I_{(C_h+1)(C_h+1)/2} \text{ for } h = 2, \ldots, H-1.$$

Note that C_1 transitions due to processing completions appear in the superdiagonal part of $Q_1^{(1)}$, implying $nz_{Q_1^{(1)}} = C_1$.

Transition H represents the self-evolution of submodel H through which parts leave the model in a way opposite to that of transition 1 so that the number of parts in the submodel decreases by 1 upon each transition. Transition H has the matrices

$$Q_H^{(H)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} & & & & \\ \mu_H & & & & \\ & \mu_H & & & \\ & & \mu_H & & \\ & & & \mu_H & \end{pmatrix} \end{array},$$

$$Q_H^{(1)} = I_{C_1+1}, \quad \text{and} \quad Q_H^{(h)} = I_{(C_h+1)(C_h+1)/2} \text{ for } h = 2, \ldots, H-1.$$

Note that C_H transitions due to processing completions appear in the subdiagonal part of $Q_H^{(H)}$, implying $nz_{Q_H^{(H)}} = C_H$.

Transitions 2 through $H-1$ represent the processing behavior of submodels 2 through $H-1$, respectively, and therefore, $Q_h^{(h)}$ has its nonzeros in the upper-triangular part. That is, for $h = 2, \ldots, H-1$, we have

$$Q_h^{(h)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \\ \begin{pmatrix} & & & & & & & & & & & & & & \\ & & & & \mu_h & & & & & & & & & & \\ & & & & & \mu_h & & & & & & & & & \\ & & & & & & \mu_h & & & & & & & & \\ & & & & & & & \mu_h & & & & & & & \\ & & & & & & & & & & & & & & \\ & & & & & & & & & \mu_h & & & & & \\ & & & & & & & & & & \mu_h & & & & \\ & & & & & & & & & & & \mu_h & & & \\ & & & & & & & & & & & & & & \\ & & & & & & & & & & & & \mu_h & & \\ & & & & & & & & & & & & & \mu_h & \\ & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & \end{pmatrix} \end{array},$$

$$Q_h^{(1)} = I_{C_1+1}, \quad Q_h^{(H)} = I_{C_H+1}\ , \quad \text{and}$$

$$Q_h^{(l)} = I_{(C_l+1)(C_l+2)/2} \ \text{ for } l \neq h,\ l \neq 1, \text{ and } l \neq H.$$

Note that $nz_{Q_h^{(h)}} = C_h(C_h+1)/2$.

We use transitions $H+1$ through $2H-1$, altogether $H-1$ transitions, to represent departure of processed parts from the hopper in each submodel and their arrival to the buffer of the machine in the next submodel in line. Since departures and arrivals in adjacent submodels need to be synchronized due to the Kanban control mechanism, each intermediate submodel h has two non-identity matrices other than $Q_h^{(h)}$, one corresponding to departures the other to arrivals. Now, we give these matrices.

Transitions $H+1$ and $2H-1$ have the matrices

$$Q_{H+1}^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} & & & & \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \end{pmatrix} \end{array}, \quad Q_{2H-1}^{(H)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \\ & & & & \end{pmatrix} \end{array},$$

$$Q_{H+1}^{(H)} = I_{C_H+1}, \quad Q_{H+1}^{(h)} = I_{(C_h+1)(C_h+1)/2} \ \text{ for } h = 3, \ldots, H-1,$$

$$Q_{2H-1}^{(1)} = I_{C_1+1}, \quad \text{and} \quad Q_{2H-1}^{(h)} = I_{(C_h+1)(C_h+1)/2} \ \text{ for } h = 2, \ldots, H-2.$$

For matrices $Q_{H+1}^{(1)}$ and $Q_{2H-1}^{(H)}$, we have $nz_{Q_{H+1}^{(1)}} = C_1$ and $nz_{Q_{2H-1}^{(H)}} = C_H$. Submodel 1 has a non-identity matrix representing departures from its output hopper, whereas submodel H has a non-identity matrix representing arrivals to the buffer of its machine. Note that arrivals are possible only when the next submodel in line has at least one unused card, whereas departures are possible only when the hopper is nonempty.

Furthermore, for each intermediate submodel h with $h = 2, \ldots, H-1$, we have the transition matrices

$$Q^{(h)}_{H+h-1} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \\ \begin{pmatrix} & 1 & & & & & & & & & & & & & \\ & & 1 & & & & & & & & & & & & \\ & & & 1 & & & & & & & & & & & \\ & & & & 1 & & & & & & & & & & \\ & & & & & & 1 & & & & & & & & \\ & & & & & & & 1 & & & & & & & \\ & & & & & & & & 1 & & & & & & \\ & & & & & & & & & 1 & & & & & \\ & & & & & & & & & & 1 & & & & \\ & & & & & & & & & & & 1 & & & \\ & & & & & & & & & & & & 1 & & \\ & & & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & & & \end{pmatrix} \end{array},$$

$$Q^{(h)}_{H+h} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \\ \begin{pmatrix} & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & \\ 1 & & & & & & & & & & & & & & \\ & 1 & & & & & & & & & & & & & \\ & & 1 & & & & & & & & & & & & \\ & & & 1 & & & & & & & & & & & \\ & & & & & 1 & & & & & & & & & \\ & & & & & & 1 & & & & & & & & \\ & & & & & & & 1 & & & & & & & \\ & & & & & & & & 1 & & & & & & \\ & & & & & & & & & 1 & & & & & \\ & & & & & & & & & & 1 & & & & \\ & & & & & & & & & & & 1 & & & \\ & & & & & & & & & & & & & & \end{pmatrix} \end{array},$$

$$Q^{(h)}_{H+l} = I_{(C_h+1)(C_h+2)/2} \text{ for } l \neq h-1, l \neq h, \text{ and } 2 \leqslant l \leqslant H-1.$$

Note that $nz_{Q^{(h)}_{H+h-1}} = nz_{Q^{(h)}_{H+h}} = C_h(C_h+1)/2$.

Finally, rates of transitions 1 through $2H-1$ are given by

$$(\alpha_1, \ldots, \alpha_H, \alpha_{H+1}, \ldots, \alpha_{2H-1}) = (1, \ldots, 1, \gamma_1, \ldots, \gamma_{H-1}) .$$

Kronecker representation of the underlying generator matrix for the production line model may then be written as

$$Q = \sum_{k=1}^{2H-1} \alpha_k \bigotimes_{h=1}^{H} Q_k^{(h)} + Q_D, \quad Q_D = -\sum_{k=1}^{2H-1} \alpha_k \bigotimes_{h=1}^{H} \operatorname{diag}(Q_k^{(h)}\mathbf{e}) .$$

The number of floating-point values to be stored in this representation is $2H-1$ for the rates, $C_1 + C_H + \sum_{h=2}^{H-1} C_h(C_h+1)/2$ for the local transition matrices, and $C_1 + C_H + 2\sum_{h=2}^{H-1} C_h(C_h+1)/2$ for the synchronizing transition matrices, thus altogether,

$$2H - 1 + 2(C_1 + C_H) + 1.5 \sum_{h=2}^{H-1} C_h(C_h+1).$$

For $H = 4$ and $C_1 = C_2 = C_3 = C_4 = 4$, this evaluates to 83, whereas the same number is 211 for $H = 8$ and $C_h = 4$ for $h = 1, \ldots, H$.

For the calculation of the number of nonzeros in the off-diagonal part of Q, let us consider the following derivation. Transitions 1 through H altogether contribute

$$(2C_1C_H + C_1 + C_H) \prod_{h=2}^{H-1} \frac{(C_h+1)(C_h+2)}{2}$$
$$+ (C_1+1)(C_H+1) \sum_{l=2}^{H-1} \frac{C_l(C_l+1)}{2} \prod_{\substack{h=2 \\ h\neq l}}^{H-1} \frac{(C_h+1)(C_h+2)}{2}$$

nonzeros to the off-diagonal part of Q. On the other hand, transitions $H+1$ through $2H-1$ altogether contribute

$$C_1(C_H+1)\frac{C_2(C_2+1)}{2} \prod_{h=3}^{H-1} \frac{(C_h+1)(C_h+2)}{2}$$
$$+ (C_1+1)(C_H+1) \sum_{l=2}^{H-1} \frac{C_l(C_l+1)}{2} \frac{C_{l+1}(C_{l+1}+1)}{2} \prod_{\substack{h=2 \\ h\neq l, h\neq l+1}}^{H-1} \frac{(C_h+1)(C_h+2)}{2}$$
$$+ (C_1+1)C_H \frac{C_{H-1}(C_{H-1}+1)}{2} \prod_{h=2}^{H-2} \frac{(C_h+1)(C_h+2)}{2}$$

nonzeros to the off-diagonal part of Q.

When $C_h = C$ for $h = 1, \ldots, H$ with $H \geqslant 4$, the contribution from transitions 1 through H becomes

$$\frac{C^2(C+1)^{H-1}(C+2)^{H-3}}{2^{H-3}},$$

and the contribution from transitions $H+1$ through $2H-1$ becomes

$$\frac{(H-3)C^2(C+1)^H(C+2)^{H-4}}{2^{H-2}}.$$

After summing these up, we obtain

$$\frac{C^2(C+1)^{H-1}(C+2)^{H-4}(HC-C+H+1)}{2^{H-2}}$$

as the number of nonzeros in the off-diagonal part of Q when $C_h = C$ for $h = 1, \ldots, H$ with $H \geqslant 4$.

When $H = 4$ and $C_1 = C_2 = C_3 = C_4 = 4$, Q is of order 5,625, and there are 8,500 nonzeros in its off-diagonal part, whereas when $H = 8$ and $C_h = 4$ for $h = 1, \ldots, H$, Q is of order 284,765,625 and has a total of 936,562,500 nonzero off-diagonal entries.

3.3.2 A Communications Protocol Model

In this subsection, we look into a communications protocol model implemented in software [328]. The relatively complex model of four submodels represents the flow of messages from a sender (or source) to a receiver (or sink) over a communications network. In this protocol, which has also been used in [49, 195] as a test case, the sender side is represented using submodels 1 and 2, whereas the receiver side is represented using submodels 3 and 4 as in Figure 3.11.

A user message that emerges in submodel 1 at the sender through t_1 is passed to the session layer through t_3 which in turn passes the message to the transport layer through t_6. The transfer of messages between adjacent layers is handled by Courier tasks `courier1` and `courier2`, where each Courier task is an active buffer taking care of the give and take of a single message. Once a message arrives in submodel 2 at the transport layer of the sender in the form of a transport service data unit (TSDU) (i.e., `transp_exec_TSDU`), it is broken down through t_7 into $1 + \omega_8/\omega_9$ transport protocol data units (TPDUs) or packets. All packets except the last one move through t_8, whereas the last packet takes the path through t_9. For each packet that is transmitted from the sender to the receiver over the communications network through t_{18} or t_{19}, an acknowledgment is generated in submodel 3 at the transport layer of the receiver through t_{21} and returned to the sender over the network through t_{20}. However, reassembly of the TSDU and its delivery to the session layer of the receiver in submodel 4 can only be performed when the last packet belonging to the TSDU arrives at the receiver through t_{18} and is processed through t_{25}. Similar operations, though in reverse order of those in submodel 1, are executed once the reassembled TSDU is passed from the transport layer through t_{28} to the session layer at the receiver in submodel 4.

The `window size` associated with packets at the sender in this model, in other words, the maximum number of unacknowledged packets sent over the network to the receiver at any given time, is C. The transport buffer space at the sender (i.e., `transp_space`) is set to 1, meaning a new message can

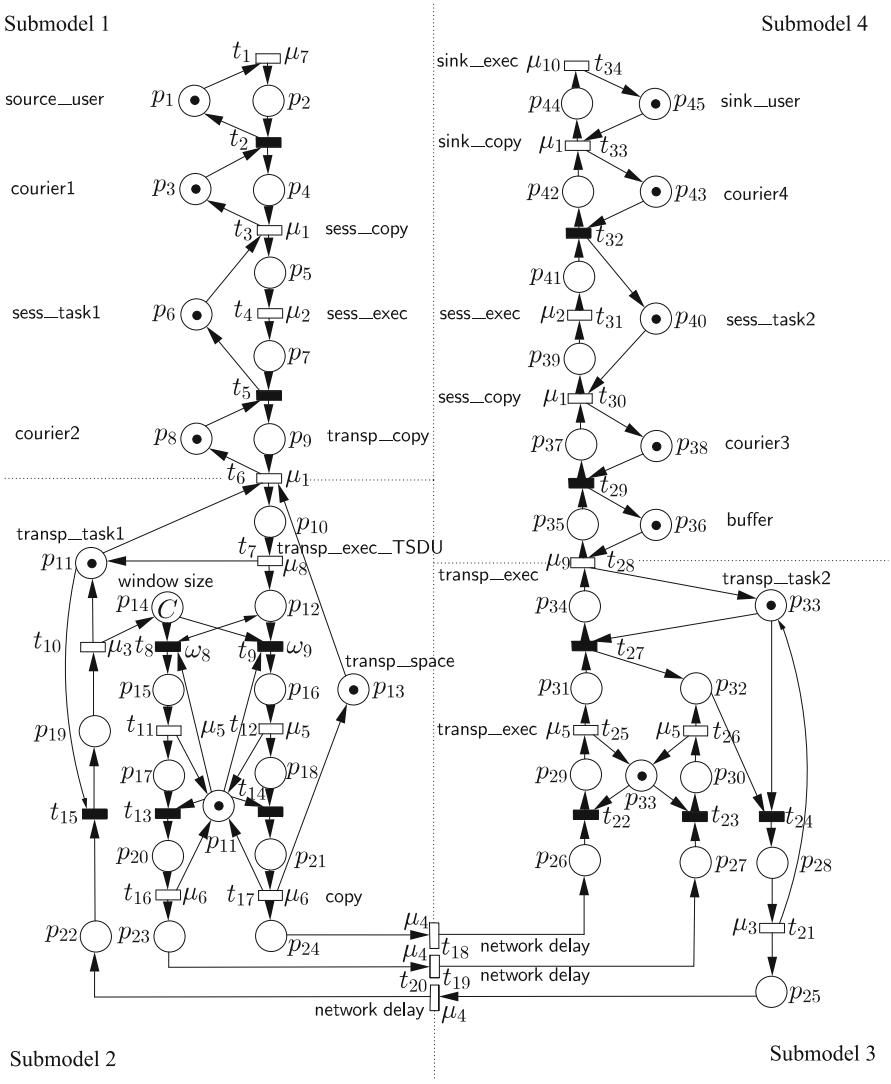


Fig. 3.11 Communications protocol model with four submodels

only be admitted to the transport layer in submodel 2 from the session layer in submodel 1 through t_6 if the last packet belonging to the TSDU at hand is ready to be transmitted to the receiver.

The *generalized stochastic Petri net* (GSPN) model of the communications protocol depicted in Figure 3.11 is a fusion of the GSPN models in Figure 3.2 of [328] and Figure 2 of [195]. The figure may seem dauntingly complex at first even though it only corresponds to a model with $H = 4$ submodels. Let us briefly recall GSPNs so that we can follow the operation of the protocol in more detail.

Formally, a GSPN [2] is an eight-tuple

$$GSPN = (\mathcal{P}, \mathcal{T}, \mathbf{pr}, \mathcal{I}, \mathcal{O}, \mathcal{H}, \mathbf{m}_0, \boldsymbol{\omega}),$$

where $\mathcal{P}$ is the set of places, $\mathcal{T}$ is the set of transitions, $\mathbf{pr}$ is the vector of *priorities* assigned to transitions in $\mathcal{T}$, $\mathcal{I}$ is the set of input arcs such that $\mathcal{I} \subset \mathcal{P} \times \mathcal{T}$, $\mathcal{O}$ is the set of output arcs such that $\mathcal{O} \subset \mathcal{T} \times \mathcal{P}$, $\mathcal{H}$ is the set of *inhibitor arcs* such that $\mathcal{H} \subset \mathcal{P} \times \mathcal{T}$, $\mathbf{m}_0$ is the *initial marking* associated with the places in $\mathcal{P}$, and $\boldsymbol{\omega}$ is a vector comprised of firing rates for *timed transitions* and weights used in the computation of firing probabilities for *immediate transitions* in $\mathcal{T}$.

In the graphical representation of the GSPN in Figure 3.11, places are indicated using circles, timed transitions are indicated using white rectangular boxes, and immediate transitions are indicated using black segments. Tokens inside places are indicated using black dots except p_{14} which initially has C tokens. Note that all arcs are directed, the directions being shown by arrowheads. For the particular model, $|\mathcal{P}| = 45$ with, respectively, 9, 15, 10, and 11 places in submodels 1, 2, 3, and 4, and $|\mathcal{T}| = 34$ with 21 timed and 13 immediate transitions. The initial markings (or states) associated with the submodels are given by

$$\begin{aligned}
\mathbf{m}_0^{(1)} &= (1, 0, 1, 0, 0, 1, 0, 1, 0)\ , \\
\mathbf{m}_0^{(2)} &= (0, 1, 0, 1, C, 0, 0, 0, 0, 0, 0, 0, 0, 0)\ , \\
\mathbf{m}_0^{(3)} &= (0, 0, 0, 0, 0, 0, 0, 0, 1, 0)\ , \\
\mathbf{m}_0^{(4)} &= (0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1)\ .
\end{aligned}$$

We have numbered the transition rates as in [328] but have adopted the numbering of places and transitions in [195]. As in [195], we have drawn each of the places p_{11} and p_{33} in Figure 3.11 twice so as to have an uncluttered graph without intersecting arcs.

In order for a transition to be enabled and thus fire in a particular marking, each of its input places must have at least as many tokens as indicated by the *multiplicity* of its corresponding input arc. By default, a multiplicity of 1 is assumed for each arc. For instance, transition t_3 fires when p_4 and p_6 are both nonzero in a marking. On the other hand, p_{11}, p_{12}, and p_{14} all need to be nonzero for t_8 or t_9 to fire. This brings us to the concept of inhibitor arcs which are indicated by small circles instead of arrowheads on the arcs. An inhibitor arc disables the transition in any marking where the number of tokens in its input place is greater than or equal to the multiplicity of the inhibitor arc. Observe that there are no inhibitor arcs in Figure 3.11; hence, $\mathcal{H} = \varnothing$.

Starting from an initial marking $\mathbf{m}_0$, the model moves into a new marking by following one of its enabled, that is, uninhibited transitions. It is assumed that transitions can have delays that are exponentially distributed with a

given firing rate or they can be immediate, meaning they take place in zero time once they are enabled. The former kind of transitions are referred to as being *timed* and are assigned the lowest priority of 0, while *immediate transitions* are assigned higher priorities starting from 1. Priorities are assumed to be the same among immediate transitions in Figure 3.11 where, for instance, t_1 is a timed transition and t_2 is an immediate transition.

When an enabled transition fires in a marking $\mathbf{m}$, tokens are removed from its input places and put in its output places. The number of tokens transferred as such from input places to output places of a transition depends on the multiplicities associated with the input arcs and output arcs of the transition. In the case of immediate transitions, the selection of which enabled transition to fire in marking $\mathbf{m}$ is based on the values of priorities in $\mathbf{pr}$ and weights in $\boldsymbol{\omega}$. If there is a tie among priorities, the selection is probabilistic and governed by

$$Prob(t_k) = \frac{\omega_k}{\sum_{\forall l \text{ such that } t_l \in \mathcal{E}(\mathbf{m})} \omega_l}$$

for enabled immediate transition t_k with the highest priority, where $\mathcal{E}(\mathbf{m})$ is the set of enabled immediate transitions with highest priority in marking $\mathbf{m}$.

Markings that enable timed transitions only are named *tangible* and comprise the *reachable state space* $\mathcal{R}$ of the underlying CTMC. On the other hand, markings that enable at least one immediate transition are named *vanishing* (or *intangible*) since they are occupied for zero time due to the priority of immediate transitions over timed transitions. For instance, in Figure 3.11 the initial marking $\mathbf{m}_0^{(1)} = (1,0,1,0,0,1,0,1,0)$ in submodel 1 is tangible since timed transition t_1 is the only enabled transition that can fire when in $\mathbf{m}_0^{(1)}$, whereas the marking $\mathbf{m}^{(1)} = (0,1,1,0,0,1,0,1,0)$ which is obtained from $\mathbf{m}_0^{(1)}$ when t_1 fires is vanishing, since $\mathbf{m}^{(1)}$ has $p_2 = p_3 = 1$ that will cause immediate transition t_2 to fire in zero time taking $\mathbf{m}^{(1)}$ to the tangible marking $\mathbf{m}_1^{(1)} = (1,0,0,1,0,1,0,1,0)$. As we will see, the CTMC underlying the GSPN becomes considerably large, and the enumeration of its reachable states (i.e., tangible markings) is manually intractable as the value of C increases.

Before moving to the state space descriptions of submodels, we recall the values of the reciprocals of the transition rates that are reported in [328] as

$$(\mu_1^{-1}, \mu_2^{-1}, \mu_3^{-1}, \mu_4^{-1}, \mu_5^{-1}, \mu_6^{-1}, \mu_7^{-1}, \mu_8^{-1}, \mu_9^{-1}, \mu_{10}^{-1}) =$$
$$(0.57, 4.97, 1.09, 10.37, 4.29, 0.39, 0.68, 2.88, 3.45, 1.25)/5{,}000\ .$$

The weights in Figure 3.11 for all immediate transitions can be considered to be 1 except t_8 and t_9 which are competing with each other in submodel 2 when they are both enabled. For t_8 and t_9 to be both enabled in a marking, p_{11}, p_{12}, and p_{14} must all be nonzero as mentioned before. We remark that t_8 is at the start of the path that takes all packets belonging to a TSDU

except the last one over the network to transition t_{26} in submodel 3. On the other hand, t_9 is at the start of the path that takes the last packet of a TSDU over the network to transition t_{25} in submodel 3. Since transitions t_8 and t_9 have weights of ω_8 and ω_9, they fire, respectively, with probabilities of $\omega_8/(\omega_8+\omega_9)$ and $\omega_9/(\omega_8+\omega_9)$ when they are both enabled. Observe that the arc between t_8 and p_{12} is bidirectional meaning this arc is both an input arc and an output arc for t_8. However, this is not the case for the arc between p_{12} and t_9. This suggests a nonsymmetric firing relationship between t_8 and t_9 once t_9 is fired.

Submodel 1 has 15 states (i.e., $|\mathcal{S}^{(1)}| = 15$) one of which is $\mathbf{m}_0^{(1)}$. The initial marking $\mathbf{m}_0^{(1)}$ corresponds to an empty submodel with no messages and is assigned state number 0. There are three states in submodel 1 that have a single message. These three states differ from each other in the location of the message among the places p_4, p_5, and p_9, which are all followed by timed transitions and are, respectively, numbered 1, 2, and 3 with the corresponding markings

$$\mathbf{m}_1^{(1)} = (1,0,0,1,0,1,0,1,0)\ , \quad \mathbf{m}_2^{(1)} = (1,0,1,0,1,0,0,1,0)\ ,$$
$$\mathbf{m}_3^{(1)} = (1,0,1,0,0,1,0,0,1)\ .$$

Submodel 1 has five states in which there are two messages. In these states which are numbered 4 through 8, the two messages are in the pairs of places (p_2,p_4), (p_4,p_5), (p_4,p_9), (p_5,p_9), and (p_7,p_9). Note that in each pair the second place is followed by a timed transition and the first place is followed by either a timed transition or an immediate transition with an empty input place that inhibits firing of the timed transition with $p_3 = 0$ as in the pair (p_2,p_4) or with $p_8 = 0$ as in the pair (p_7,p_9). The associated markings are

$$\mathbf{m}_4^{(1)} = (0,1,0,1,0,1,0,1,0)\ , \quad \mathbf{m}_5^{(1)} = (1,0,0,1,1,0,0,1,0)\ ,$$
$$\mathbf{m}_6^{(1)} = (1,0,0,1,0,1,0,0,1)\ , \quad \mathbf{m}_7^{(1)} = (1,0,1,0,1,0,0,0,1)\ ,$$
$$\mathbf{m}_8^{(1)} = (1,0,1,0,0,0,1,0,1)\ .$$

Submodel 1 has four states in which there are three messages in the triples of places (p_2,p_4,p_5), (p_2,p_4,p_9), (p_4,p_5,p_9), and (p_4,p_7,p_9). These states are numbered 9 through 12 and have the corresponding markings

$$\mathbf{m}_9^{(1)} = (0,1,0,1,1,0,0,1,0)\ , \quad \mathbf{m}_{10}^{(1)} = (0,1,0,1,0,1,0,0,1)\ ,$$
$$\mathbf{m}_{11}^{(1)} = (1,0,0,1,1,0,0,0,1)\ , \quad \mathbf{m}_{12}^{(1)} = (1,0,0,1,0,0,1,0,1)\ .$$

Finally, there are two states in which submodel 1 has four messages. These states are numbered 13 and 14 and are given by the markings

$$\mathbf{m}_{13}^{(1)} = (0,1,0,1,1,0,0,0,1)\ , \quad \mathbf{m}_{14}^{(1)} = (0,1,0,1,0,0,1,0,1)\ .$$

Note that it is not possible to have five or more messages in submodel 1 since the total number of tokens in its initial marking is four and this number is preserved in all markings due to the structure of the GSPN. The number of states in submodel 1 is independent of C and so is the one in submodel 4. That is why we choose to discuss submodel 4 next.

Submodel 4 has 30 states (i.e., $|\mathcal{S}^{(4)}| = 30$), twice that of submodel 1 and one of which is $\mathbf{m}_0^{(4)}$. As in submodel 1, the initial marking $\mathbf{m}_0^{(4)}$ also represents an empty submodel with no messages and is assigned state number 0. There are four states in submodel 4 that have a single message. These four states differ from each other in the location of the message among the places p_{37}, p_{39}, p_{42}, and p_{44} and are, respectively, numbered 1 through 4 with the markings

$$\mathbf{m}_1^{(4)} = (0,1,1,0,0,1,0,0,1,0,1)\ , \quad \mathbf{m}_2^{(4)} = (0,1,0,1,1,0,0,0,1,0,1)\ ,$$
$$\mathbf{m}_3^{(4)} = (0,1,0,1,0,1,0,1,0,0,1)\ , \quad \mathbf{m}_4^{(4)} = (0,1,0,1,0,1,0,0,1,1,0)\ .$$

Submodel 4 has eight states in which there are two messages. In these states which are numbered 5 through 12, the two messages are in the pairs of places (p_{35}, p_{37}), (p_{37}, p_{39}), (p_{37}, p_{42}), (p_{37}, p_{44}), (p_{39}, p_{42}), (p_{39}, p_{44}), (p_{41}, p_{42}), and (p_{42}, p_{44}). The corresponding markings are

$$\mathbf{m}_5^{(4)} = (1,0,1,0,0,1,0,0,1,0,1)\ , \quad \mathbf{m}_6^{(4)} = (0,1,1,0,1,0,0,0,1,0,1)\ ,$$
$$\mathbf{m}_7^{(4)} = (0,1,1,0,0,1,0,1,0,0,1)\ , \quad \mathbf{m}_8^{(4)} = (0,1,1,0,0,1,0,0,1,1,0)\ ,$$
$$\mathbf{m}_9^{(4)} = (0,1,0,1,1,0,0,1,0,0,1)\ , \quad \mathbf{m}_{10}^{(4)} = (0,1,0,1,1,0,0,0,1,1,0)\ ,$$
$$\mathbf{m}_{11}^{(4)} = (0,1,0,1,0,0,1,1,0,0,1)\ , \quad \mathbf{m}_{12}^{(4)} = (0,1,0,1,0,1,0,1,0,1,0)\ .$$

Submodel 4 has nine states in which there are three messages in the triples of places (p_{35}, p_{37}, p_{39}), (p_{35}, p_{37}, p_{42}), (p_{35}, p_{37}, p_{44}), (p_{37}, p_{39}, p_{42}), (p_{37}, p_{39}, p_{44}), (p_{37}, p_{41}, p_{42}), (p_{37}, p_{42}, p_{44}), (p_{39}, p_{42}, p_{44}), and (p_{41}, p_{42}, p_{44}). These states numbered 13 through 21 have the corresponding markings

$$\mathbf{m}_{13}^{(4)} = (1,0,1,0,1,0,0,0,1,0,1)\ , \quad \mathbf{m}_{14}^{(4)} = (1,0,1,0,0,1,0,1,0,0,1)\ ,$$
$$\mathbf{m}_{15}^{(4)} = (1,0,1,0,0,1,0,0,1,1,0)\ , \quad \mathbf{m}_{16}^{(4)} = (0,1,1,0,1,0,0,1,0,0,1)\ ,$$
$$\mathbf{m}_{17}^{(4)} = (0,1,1,0,1,0,0,0,1,1,0)\ , \quad \mathbf{m}_{18}^{(4)} = (0,1,1,0,0,0,1,1,0,0,1)\ ,$$
$$\mathbf{m}_{19}^{(4)} = (0,1,1,0,0,1,0,1,0,1,0)\ , \quad \mathbf{m}_{20}^{(4)} = (0,1,0,1,1,0,0,1,0,1,0)\ ,$$
$$\mathbf{m}_{21}^{(4)} = (0,1,0,1,0,0,1,1,0,1,0)\ .$$

Submodel 4 has six states in which there are four messages in the quadruples of places $(p_{35}, p_{37}, p_{39}, p_{42})$, $(p_{35}, p_{37}, p_{39}, p_{44})$, $(p_{35}, p_{37}, p_{41}, p_{42})$, $(p_{35}, p_{37}, p_{42}, p_{44})$, $(p_{37}, p_{39}, p_{42}, p_{44})$, and $(p_{37}, p_{41}, p_{42}, p_{44})$. These states are numbered 22 through 27 with the corresponding markings

$$\mathbf{m}_{22}^{(4)} = (1,0,1,0,1,0,0,1,0,0,1)\ , \quad \mathbf{m}_{23}^{(4)} = (1,0,1,0,1,0,0,0,1,1,0)\ ,$$
$$\mathbf{m}_{24}^{(4)} = (1,0,1,0,0,0,1,1,0,0,1)\ , \quad \mathbf{m}_{25}^{(4)} = (1,0,1,0,0,1,0,1,0,1,0)\ ,$$
$$\mathbf{m}_{26}^{(4)} = (0,1,1,0,1,0,0,1,0,1,0)\ , \quad \mathbf{m}_{27}^{(4)} = (0,1,1,0,0,0,1,1,0,1,0)\ .$$

Finally, there are two states in which submodel 4 has five messages. These states are numbered 28 and 29 and are given by the markings

$$\mathbf{m}_{28}^{(4)} = (1,0,1,0,1,0,0,1,0,1,0), \quad \mathbf{m}_{29}^{(4)} = (1,0,1,0,0,0,1,1,0,1,0)\ .$$

Note that it is not possible to have six or more messages in submodel 4 since the total number of tokens in its initial marking is five and this number is preserved in all markings due to the structure of the GSPN.

Now, we turn to submodels 2 and 3, which have state space sizes that depend on the value of C. For $C = 1$, submodel 2 has 16 states (i.e., $|\mathcal{S}^{(2)}| = 16$), and submodel 3 has 6 states (i.e., $|\mathcal{S}^{(3)}| = 6$). In the following, we let $C = 1$ and $\omega_8 = \omega_9 = 1$, meaning the `window size` is set to 1 and each TSDU is broken down into 2 packets at the transport layer. The particular values of ω_8 and ω_9 suggest that immediate transitions t_8 and t_9 each fire with a probability of 1/2 when they are both enabled. In fact, p_{11} is at the source of input arcs of the five immediate transitions t_8, t_9, t_{13}, t_{14}, and t_{15} in submodel 2. Similarly, p_{33} is at the source of input arcs of the four immediate transitions t_{22}, t_{23}, t_{24}, and t_{27} in submodel 3. For instance, when $C = 2$, there will be firing probabilities of 1/3 in submodel 2 and 1/2 in submodel 3, which do not exist for $C = 1$, even though all immediate transitions in the GSPN still have weights of 1. The reason behind this is the existence of vanishing markings in these two submodels which, respectively, have three and two immediate transitions that are enabled in this case. This is something to reckon with for larger values of C.

We first consider the simpler submodel 3. The initial marking $\mathbf{m}_0^{(3)}$ corresponds to an empty submodel with no TSDU's and is assigned state number 0. The markings associated with states numbered 1 through 5 are

$$\mathbf{m}_1^{(3)} = (0,0,0,0,0,1,0,0,0,0)\ , \quad \mathbf{m}_2^{(3)} = (0,0,0,1,0,0,0,0,0,0)\ ,$$
$$\mathbf{m}_3^{(3)} = (1,0,0,0,0,0,0,0,1,0)\ , \quad \mathbf{m}_4^{(3)} = (0,0,0,0,1,0,0,0,0,0)\ ,$$
$$\mathbf{m}_5^{(3)} = (0,0,0,0,0,0,0,1,0,1)\ .$$

State 1 corresponds to having the first packet of a TSDU in submodel 3 since $p_{30} = 1$. State 2 corresponds to getting an acknowledgment packet prepared since $p_{28} = 1$. Note that this acknowledgment could be for the first, or the second (hence last) packet of a TSDU. State 3 corresponds to having an acknowledgment ready to leave submodel 3 since $p_{25} = 1$. State 4 corresponds to having the second packet in submodel 3 since $p_{29} = 1$. Finally, state 5 corresponds to having a reassembled TSDU ready to be passed to the

session layer in submodel 4 since $p_{34} = 1$ and an acknowledgment packet on its way to being prepared since $p_{32} = 1$.

Submodel 2 is more complicated than submodel 3 in its operation. As in submodel 3, the initial marking $\mathbf{m}_0^{(2)}$ corresponds to an empty submodel with no TSDU's and is assigned state number 0. The markings associated with states numbered 1 through 15 are

$$\begin{aligned}
\mathbf{m}_1^{(2)} &= (1,0,0,0,1,0,0,0,0,0,0,0,0,0,0)\ , \\
\mathbf{m}_2^{(2)} &= (0,0,1,0,0,1,0,0,0,0,0,0,0,0,0)\ , \\
\mathbf{m}_3^{(2)} &= (0,0,1,0,0,0,0,0,0,0,1,0,0,0,0)\ , \\
\mathbf{m}_4^{(2)} &= (0,1,1,0,0,0,0,0,0,0,0,0,0,1,0)\ , \\
\mathbf{m}_5^{(2)} &= (0,0,1,0,0,0,0,0,0,1,0,0,0,0,0)\ , \\
\mathbf{m}_6^{(2)} &= (0,0,0,0,0,0,1,0,0,0,0,0,0,0,0)\ , \\
\mathbf{m}_7^{(2)} &= (0,0,0,0,0,0,0,0,0,0,0,1,0,0,0)\ , \\
\mathbf{m}_8^{(2)} &= (0,1,0,1,0,0,0,0,0,0,0,0,0,0,1)\ , \\
\mathbf{m}_9^{(2)} &= (0,0,0,1,0,0,0,0,0,1,0,0,0,0,0)\ , \\
\mathbf{m}_{10}^{(2)} &= (1,0,0,0,0,0,0,0,0,0,0,0,0,0,1)\ , \\
\mathbf{m}_{11}^{(2)} &= (1,0,0,0,0,0,0,0,0,0,0,0,1,0,0)\ , \\
\mathbf{m}_{12}^{(2)} &= (0,1,1,0,0,0,0,0,0,0,0,0,0,0,1)\ , \\
\mathbf{m}_{13}^{(2)} &= (0,1,1,0,0,0,0,0,0,0,0,0,0,0,0)\ , \\
\mathbf{m}_{14}^{(2)} &= (0,1,0,1,0,0,0,0,0,0,0,0,0,0,0)\ , \\
\mathbf{m}_{15}^{(2)} &= (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)\ .
\end{aligned}$$

State 1 corresponds to having a TSDU in submodel 2 since $p_{10} = 1$. States 2, 3, 4, and 5 with $p_{12} = 1$ correspond to having a first packet, preparing a first packet, having a ready to leave first packet, and having its acknowledgment arrival in submodel 2 since $p_{15} = 1$, $p_{20} = 1$, $p_{11} = p_{23} = 1$, and $p_{19} = 1$, respectively. States 6, 7, 8, and 9 with $p_{12} = 0$ correspond to having a last packet, preparing a last packet, having a ready to leave last packet, and having its acknowledgment arrival in submodel 2 since $p_{16} = 1$, $p_{21} = 1$, $p_{11} = p_{13} = p_{24} = 1$, and $p_{13} = p_{19} = 1$, respectively. State 10 corresponds to having a new TSDU and a ready to leave last packet since $p_{10} = p_{24} = 1$, and state 11 corresponds to having a new TSDU and an acknowledgment arrival since $p_{10} = p_{22} = 1$. State 12 is a state similar to state 10, but one in which the TSDU has moved to p_{12} from p_{10}. Note that the TSDU cannot move further to p_{15} or p_{16} and generate a packet because $p_{14} = 0$. State 13 with $p_{11} = p_{12} = 1$ and state 14 with $p_{11} = p_{13} = 1$ correspond to having a departed first packet and a departed last packet, respectively. State 15 corresponds to having a new TSDU and a departed last packet since all places are 0 except p_{10}.

Reachable state space generation in the context of GSPNs is performed by starting from a state corresponding to a tangible initial marking and generating with breadth-first search [84] all successor states that correspond to tangible markings for each unprocessed state until no more states can be generated. Now, recall that submodels 1 through 4, respectively, have 9, 15, 10, and 11 places each of which can hold at most one token for $C = 1$, implying product state spaces of 2^9, 2^{15}, 2^{10}, and 2^{11}. However, we have shown that sizes of these submodel reachable state spaces are given by $|\mathcal{S}^{(1)}| = 15$, $|\mathcal{S}^{(2)}| = 16$, $|\mathcal{S}^{(3)}| = 6$, and $|\mathcal{S}^{(4)}| = 30$. Clearly, each submodel's reachable state space size is much smaller than its product state space size, which suggests that grouping of places and elimination of a large number of unreachable states from each product state space must have been performed. Observe that some of the states in the difference between the product state space and the reachable state space of each submodel are vanishing due to immediate transitions and therefore eliminated, while others are eliminated simply because they are not possible due to semantic constraints. With this note and that the concept of grouping will be discussed further in Chapter 4, we now turn to reachable state space $\mathcal{R}$ of the communications protocol model composed of four submodels.

In [49], an algorithm is devised to compute partitionings of submodel state spaces $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$ through which an HMM representation of an H-dimensional Markovian model expressed as a GSPN can be obtained. The input supplied to this algorithm is the GSPN model partitioned into a number of subnets, which must interact through timed transitions only. Subnets defined as such correspond to submodels in our setting. Observe that transitions t_6 between submodels 1 and 2; t_{18}, t_{19}, and t_{20} between submodels 2 and 3; and t_{28} between submodels 3 and 4 in Figure 3.11 are all timed. The automated approach for generating an HMM representation for a given GSPN model in [49] can then be viewed as consisting of the three phases described below.

In the first phase of the approach, the reachable state space of each submodel is generated by eliminating vanishing markings. This can be done locally within each submodel since by definition all immediate transitions are constrained to be inside submodels and cannot be at synchronization points between submodels. All submodel state spaces obtained as such are assumed to be finite and denoted by $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$. We have been performing the equivalent task relatively easily by manual enumeration in all the example models considered up to this point. However, in submodels 2 and 3 of the communications protocol model, it does not seem to be possible to follow the same approach for $C > 1$ due to the large number of markings involved, hence, the need for automation.

In the second phase, reachable state space $\mathcal{R}$ of the GSPN model corresponding to its tangible markings is generated and stored in a Boolean vector $\mathbf{s}$ of length $|\mathcal{S}| = \times_{h=1}^{H} |\mathcal{S}^{(h)}|$ by marking reachable states with 1 and leaving unreachable states as 0. In other words, vector $\mathbf{s}$ is of length $n = \prod_{h=1}^{H} n_h$,

where $n_h = |\mathcal{S}^{(h)}|$, and has $|\mathcal{R}|$ entries that are 1 and $n - |\mathcal{R}|$ entries that are 0. For dimension h of the H-dimensional model, Boolean vector $\mathbf{s}$ is treated as a matrix which has the states in $\mathcal{S}^{(h)}$ as row indices and states of the other $H-1$ dimensions in $\times_{l=1,l\neq h}^{H} \mathcal{S}^{(l)}$ as column indices. Then states in $\mathcal{S}^{(h)}$ are partitioned according to identical rows of this $(n_h \times n/n_h)$ Boolean matrix by placing indices of identical rows into the same partition of $\mathcal{S}^{(h)}$ under the rationale that they are reachable with the same environment states in $\times_{l=1,l\neq h}^{H} \mathcal{S}^{(l)}$. Now, let us assume that the second phase yields N_h partitions for $\mathcal{S}^{(h)}$; that is, for $h = 1, \ldots, H$, let us have

$$\mathcal{S}^{(h)} = \bigcup_{p=0}^{N_h-1} \mathcal{S}_p^{(h)}, \quad \mathcal{S}_p^{(h)} \cap \mathcal{S}_w^{(h)} = \varnothing, \ p \neq w \quad \text{for } p, w = 0, \ldots, N_h - 1. \tag{3.1}$$

The partitioning of $\mathcal{S}^{(h)}$ in this way takes $O(n)$ time and is carried out for $h = 1, \ldots, H$, thus altogether amounting to $O(Hn)$ time [49].

In our model for $C = 1$, $n = 43{,}200$ and $|\mathcal{R}| = 11{,}700$. When $\mathbf{s}$ of length n is organized into a Boolean matrix having the states in dimension h as row indices and the states in the product state space of the remaining dimensions as column indices, and after the partitioning of row indices is performed for $h = 1, \ldots, H$, we obtain $N_1 = 1$, $N_2 = 3$, $N_3 = 3$, and $N_4 = 1$. Hence, submodels 1 and 4 are comprised of single partitions of states as in

$$\mathcal{S}_0^{(1)} = \{0, \ldots, 14\} \quad \text{and} \quad \mathcal{S}_0^{(4)} = \{0, \ldots, 29\} \ ,$$

whereas submodels 2 and 3 each have three partitions of states given by

$$\mathcal{S}_0^{(2)} = \{0, \ldots, 12\}, \quad \mathcal{S}_1^{(2)} = \{13\}, \quad \mathcal{S}_2^{(2)} = \{14, 15\} \quad \text{and}$$

$$\mathcal{S}_0^{(3)} = \{0\}, \quad \mathcal{S}_1^{(3)} = \{1\}, \quad \mathcal{S}_2^{(3)} = \{2, \ldots, 5\}.$$

Observe that partition 1 of submodel 2 corresponds to the state in which the first packet of a TSDU has departed but its acknowledgment has not arrived, whereas its partition 2 corresponds to the two states in which the last packet of a TSDU has departed but its acknowledgment has not arrived. Partition 0 of submodel 2 has the remaining 13 states. When we look at the partitions of submodel 3, we note that partition 0 corresponds to the state in which the submodel is empty and partition 1 corresponds to the state in which the first packet of a TSDU from submodel 2 has arrived. Partition 2 of submodel 3 has the remaining four states.

The number of partitions, N_h, of submodel state spaces, $\mathcal{S}^{(h)}$, obtained in this way implies a total of $\prod_{h=1}^{H} N_h$ potential reachable state space partitions. When there are no unreachable states in $\mathcal{S}$, meaning $\mathcal{S} = \mathcal{R}$, we have $N_h = 1$ for $h = 1, \ldots, H$. On the other hand, when there are unreachable states in $\mathcal{S}$, meaning $\prod_{h=1}^{H} N_h > 1$, there must be some potential reachable state space partitions among the $\prod_{h=1}^{H} N_h$ that are unreachable. In fact, the minimum

number of unreachable potential reachable state space partitions must be $\max_h(N_h - 1)$ [49]. To see this, let the $\prod_{h=1}^{H} N_h$ potential reachable state space partitions be organized into an $(N_h \times \prod_{l=1,l\neq h}^{H} N_l)$ Boolean matrix. All rows of this matrix need to be different from each other; otherwise the indices of identical rows among them would have been placed into the same partition. One of the N_h rows can be all 1s, while each of the other $N_h - 1$ rows must have at least one 0 in a different position for its index to remain as it is without being placed with another row's index into the same partition. Hence, $N_h - 1$ is the minimum number of 0s in the $(N_h \times \prod_{f=1,f\neq h}^{H} N_f)$ Boolean matrix corresponding to potential reachable state space partitions for dimension h. But, this needs to be true for all H dimensions, implying we need to take the maximum of $N_h - 1$ for $h = 1, \ldots, H$.

In our model for $C = 1$, we have (1×9) and (3×3) Boolean matrices associated with potential reachable state space partitions along dimensions 1, 4 and 2, 3, respectively. For instance, the (3×3) Boolean matrix associated with dimension 2 is given by

$$\begin{array}{c} \\ \mathcal{S}_0^{(2)} \\ \mathcal{S}_1^{(2)} \\ \mathcal{S}_2^{(2)} \end{array} \begin{array}{c} \begin{array}{ccc} \mathcal{S}^{(1)} \times \mathcal{S}_0^{(3)} \times \mathcal{S}^{(4)} & \mathcal{S}^{(1)} \times \mathcal{S}_1^{(3)} \times \mathcal{S}^{(4)} & \mathcal{S}^{(1)} \times \mathcal{S}_2^{(3)} \times \mathcal{S}^{(4)} \end{array} \\ \begin{pmatrix} 1 & & \\ & 1 & 1 \\ & & 1 \end{pmatrix} \end{array}.$$

The four 1s out of the nine possible 1s in this Boolean matrix yield the $N = 4$ reachable state space partitions (i.e., macrostates) in Table 3.1. This is understandably so, because among the 96 states in $\mathcal{S}^{(2)} \times \mathcal{S}^{(3)}$, only the 26 in

$$\mathcal{S}_0^{(2)} \times \mathcal{S}_0^{(3)} \cup \mathcal{S}_1^{(2)} \times \mathcal{S}_1^{(3)} \cup \mathcal{S}_1^{(2)} \times \mathcal{S}_2^{(3)} \cup \mathcal{S}_2^{(2)} \times \mathcal{S}_2^{(3)}$$

are reachable. Therefore, we have

$$\mathcal{R} = \bigcup_{p=0}^{3} \mathcal{R}_p \quad \text{with} \quad \mathcal{R}_p = \bigtimes_{h=1}^{4} \mathcal{R}_p^{(h)} \quad \text{for } p = 0, \ldots, 3,$$

where

$$\mathcal{R}_0 = \mathcal{S}_0^{(1)} \times \mathcal{S}_0^{(2)} \times \mathcal{S}_0^{(3)} \times \mathcal{S}_0^{(4)}\,, \quad \mathcal{R}_1 = \mathcal{S}_0^{(1)} \times \mathcal{S}_1^{(2)} \times \mathcal{S}_1^{(3)} \times \mathcal{S}_0^{(4)}\,,$$
$$\mathcal{R}_2 = \mathcal{S}_0^{(1)} \times \mathcal{S}_1^{(2)} \times \mathcal{S}_2^{(3)} \times \mathcal{S}_0^{(4)}\,, \quad \mathcal{R}_3 = \mathcal{S}_0^{(1)} \times \mathcal{S}_2^{(2)} \times \mathcal{S}_2^{(3)} \times \mathcal{S}_0^{(4)}\,.$$

Table 3.1 Mapping between reachable state space partitions and submodel states in communications protocol model

Partition	Submodel 1	Submodel 2	Submodel 3	Submodel 4	# of states
$\mathcal{R}_0$	0:14	0:12	0:0	0:29	15 . 13 . 1 . 30 = 5,850
$\mathcal{R}_1$	0:14	13:13	1:1	0:29	15 . 1 . 1 . 30 = 450
$\mathcal{R}_2$	0:14	13:13	2:5	0:29	15 . 1 . 4 . 30 = 1,800
$\mathcal{R}_3$	0:14	14:15	2:5	0:29	15 . 2 . 4 . 30 = 3,600

The third phase of the approach associates timed transitions in the model between reachable state space partitions other than local transitions starting and ending in the same reachable state space partition with the entries of the $(N \times N)$ HLM matrix as in Section 2.2. Finally, the HMM representation consisting of the HLM matrix, transitions, their rates, state space partitions of submodels, and non-identity transition matrices are written to files in sparse format suitable for the `NSolve` package of the APNN toolbox [7, 22].

In our particular model, there are a total of $K = 9$ transitions. For consistency, local transition of submodel h is numbered h for $h = 1, \ldots, 4$. Transition 5 (t_6 in Figure 3.11) with rate μ_1 synchronizes submodels 1 and 2. Transitions 6, 7, and 8 (respectively, t_{18}, t_{19}, and t_{20} in Figure 3.11) with rates μ_4 synchronize submodels 2 and 3. Finally, transition 9 (t_{28} in Figure 3.11) with rate μ_9 synchronizes submodels 3 and 4.

We prefer to express the (4×4) HLM matrix using transitions numbered 1 through 9 and explicitly writing local transitions similar to what we have done with the polling model in Section 2.3 to yield the interaction matrix

$$\begin{array}{c} \\ \mathcal{R}_0 \\ \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \end{array} \begin{array}{c} \begin{array}{cccc} \mathcal{R}_0 & \mathcal{R}_1 & \mathcal{R}_2 & \mathcal{R}_3 \end{array} \\ \begin{pmatrix} \{1,2,3,4,5\} & \{7\} & \{6\} & \{6\} \\ & \{1,2,3,4\} & \{3\} & \\ \{8\} & & \{1,2,3,4,9\} & \\ \{8\} & & \{2\} & \{1,2,3,4,5,9\} \end{pmatrix} \end{array}. \tag{3.2}$$

Observe that when a local transition appears in the off-diagonal part of the interaction matrix describing the transitions among reachable state space partitions, it is associated with the submodel that changes state. We have two such examples in this model. One of them appears in entry $(\mathcal{R}_1, \mathcal{R}_2)$ where submodel 3 changes its state from 1 to another state in $\{2, \ldots, 5\}$. The other appears in entry $(\mathcal{R}_3, \mathcal{R}_2)$ where submodel 2 changes from a state in $\{14, 15\}$ to state 13. The other submodels in both cases remain in their states.

Each of the transitions 1 through 4 in (3.2) has a corresponding Kronecker product of four submodel submatrices. The submatrices associated with those submodels that do not participate in a transition are all identity. Submodels 1 and 4 each participate in two transitions, whereas submodels 2 and 3 each participate in five transitions. Submodel 1 participates in transitions $1, 5$, submodel 2 participates in transitions $2, 5, 6, 7, 8$, submodel 3 participates in transitions $3, 6, 7, 8, 9$, and submodel 4 participates in transitions $4, 9$. For instance, submodel 2 participates in transition 2 with submatrix $Q_2^{(2)}(0 : 12, 0 : 12)$ when in $\mathcal{R}_0$, with submatrix $Q_2^{(2)}(13 : 13, 13 : 13)$ when in $\mathcal{R}_1$ or $\mathcal{R}_2$, and with submatrix $Q_2^{(2)}(14 : 15, 14 : 15)$ when in $\mathcal{R}_3$. Submodel 2 participates in transition 5 with submatrix $Q_5^{(2)}(0 : 12, 0 : 12)$ when in $\mathcal{R}_0$ and with submatrix $Q_5^{(2)}(14 : 15, 14 : 15)$ when in $\mathcal{R}_3$. When in $\mathcal{R}_0$, submodel 2 participates in transition 6 with submatrix $Q_6^{(2)}(0 : 12, 13 : 13)$ if the transition is to $\mathcal{R}_2$ and with submatrix $Q_6^{(2)}(0 : 12, 14 : 15)$ if the

transition is to $\mathcal{R}_3$. Submodel 2 also participates in transition 7 when in $\mathcal{R}_0$ with submatrix $Q_7^{(2)}(0:12,13:13)$. Finally, submodel 2 participates in transition 8 with submatrix $Q_8^{(2)}(13:13,0:12)$ when in $\mathcal{R}_2$ and with submatrix $Q_8^{(2)}(14:15,0:12)$ when in $\mathcal{R}_3$.

The local transition matrix of submodel 1 is given by

$$Q_1^{(1)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \\ \begin{pmatrix} \mu_7 & & & & & & & & & & & & & & \\ & \mu_1 & & \mu_7 & & & & & & & & & & & \\ & & \mu_2 & & \mu_7 & & & & & & & & & & \\ & & & & & \mu_7 & & & & & & & & & \\ & & & & \mu_1 & & & & & & & & & & \\ & & & & & \mu_2 & & & & \mu_7 & & & & & \\ & & & & & & \mu_1 & & & & \mu_7 & & & & \\ & & & & & & & \mu_2 & & & & \mu_7 & & & \\ & & & & & & & & & & \mu_2 & & & & \\ & & & & & & & & & & & \mu_1 & & & \\ & & & & & & & & & & & & \mu_2 & \mu_7 & \\ & & & & & & & & & & & & & & \mu_7 \\ & & & & & & & & & & & & & & \mu_2 \\ & & & & & & & & & & & & & & \end{pmatrix} \end{array}.$$

Furthermore, we have $Q_1^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 2, 3, 4$.

Note that $nz_{Q_1^{(1)}} = 20$. Of the 20 nonzeros, 4 have the value μ_1, 6 have the value μ_2, and 10 have the value μ_7. In particular, the ten nonzeros with value μ_7 are in rows corresponding to states that have $p_1 = 1$ for which transition t_1 in Figure 3.11 is enabled. These nonzeros represent message arrivals to submodel 1 and therefore increase the number of messages in it by one. The four nonzeros with value μ_1 are in rows corresponding to states that have $p_4 = p_6 = 1$ for which transition t_3 in Figure 3.11 is enabled. These nonzeros represent the passing of a message to the session layer. On the other hand, the six nonzeros with value μ_2 are in rows corresponding to states that have $p_5 = 1$ for which transition t_4 in Figure 3.11 is enabled. These nonzeros represent the processing of the message at the session layer.

The local transition matrix of submodel 2 is given by

$$Q_2^{(2)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \\ \left(\begin{array}{cccccccccccccccc} & & 0.5\mu_8 & & & & 0.5\mu_8 & & & & & & & & & \\ & & & \mu_5 & & & & & & & & & & & & \\ & & & & \mu_6 & & & & & & & & & & & \\ & & 0.5\mu_3 & & & & 0.5\mu_3 & & & & & & & & & \\ & & & & & & & \mu_5 & & & & & & & & \\ & & & & & & & & \mu_6 & & & & & & & \\ \mu_3 & & & & & & & & & & & & \mu_8 & & & \\ & & & & & \mu_8 & & & & & & & & & & \\ \hline \hline & & & & & & & & & & & & & & & \mu_8 \end{array} \right) \end{array}.$$

Furthermore, we have $Q_2^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 1, 3, 4$.

Note that $nz_{Q_2^{(2)}} = 12$. Of the 12 nonzeros, 3, 2, 2, and 5 assume values that, respectively, depend on μ_3, μ_5, μ_6, and μ_8. In particular, the three nonzeros with values depending on μ_3 are in rows corresponding to states that have $p_{19} = 1$ for which transition t_{10} in Figure 3.11 is enabled. These nonzeros represent acknowledgment packet processing at the transport layer in submodel 2 and therefore increase the `window size` in p_{14} by one and set $p_{11} = 1$. When in state 5, which has $p_{12} = 1$, this implies that immediate transitions t_8 and t_9 in Figure 3.11 both become enabled. Each of these two transitions fire with a probability of 0.5 as mentioned before; hence, the submodel immediately moves to either state 2 in which the first packet or state 6 in which the second packet of a TSDU is to be formed. That is why there are two entries in row 5 with values $0.5\mu_3$. The two nonzeros with values μ_5 are in rows corresponding to states that have $p_{15} = 1$ for which transition t_{11} and $p_{16} = 1$ for which transition t_{12} in Figure 3.11 are enabled. These nonzeros represent first and second packet preparation at the transport layer in submodel 2 and set $p_{17} = 1$ and $p_{18} = 1$, respectively. The two nonzeros with values μ_6 are in rows corresponding to states that have $p_{20} = 1$ for which transition t_{16} and $p_{21} = 1$ for which transition t_{17} in Figure 3.11 are enabled. These nonzeros, respectively, represent a first or a second packet becoming ready to leave the transport layer in submodel 2 and move over the network to the transport layer of submodel 3. They, respectively, set $p_{23} = 1$ and $p_{13} = p_{24} = 1$. On the other hand, the five nonzeros with values depending on μ_8 are in rows corresponding to states that have $p_{10} = 1$ for which transition t_7 in Figure 3.11 is enabled. These nonzeros correspond to the processing of a TSDU at the transport layer in submodel 2 and set $p_{12} = 1$. When in

state 1, which has $p_{10} = p_{14} = 1$, this implies that transitions t_8 and t_9 in Figure 3.11 both become enabled. But, we have already discussed this for row 5 and explained the existence of two entries. The same applies here, yet with values of $0.5\mu_8$. Finally, we remark that the nonzero with value μ_8 in entry (15,13) is in an off-diagonal block and corresponds to transition 2 in entry $(\mathcal{R}_3, \mathcal{R}_2)$ of (3.2).

The local transition matrix of submodel 3 is given by

$$Q_3^{(3)} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \left(\begin{array}{cccccc} & & & & & \\ & & \mu_5 & & & \\ & & & \mu_3 & & \\ & & & & & \\ & & & & & \mu_5 \\ & & & & & \end{array} \right) \end{array}$$

with $nz_{Q_3^{(3)}} = 3$. The two nonzeros with value μ_5 in $Q_3^{(3)}$ correspond to the processing of first and second packets of a TSDU at the transport layer in submodel 3, respectively, through transitions t_{26} and t_{25} in Figure 3.11. We remark that the nonzero with value μ_5 in entry (1,2) is in an off-diagonal block and corresponds to transition 3 in entry $(\mathcal{R}_1, \mathcal{R}_2)$ of (3.2). The nonzero with value μ_3 corresponds to the preparation of an acknowledgment packet through transition t_{21} in Figure 3.11 and becoming ready to leave submodel 3. Furthermore, we have $Q_3^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 1, 2, 4$.

Since submodel 4 has 30 states and $nz_{Q_4^{(4)}} = 44$, rather than providing $Q_4^{(4)}$ in matrix form, we list its 44 nonzero entries in three groups. Note that $Q_4^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 1, 2, 3$. The first group has the 15 nonzeros

$$\begin{aligned} & q_4^{(4)}(4,0) = q_4^{(4)}(8,1) = q_4^{(4)}(10,2) = q_4^{(4)}(12,3) \\ = \; & q_4^{(4)}(15,5) = q_4^{(4)}(17,6) = q_4^{(4)}(19,7) = q_4^{(4)}(20,9) \\ = \; & q_4^{(4)}(21,11) = q_4^{(4)}(23,13) = q_4^{(4)}(25,14) = q_4^{(4)}(26,16) \\ = \; & q_4^{(4)}(27,18) = q_4^{(4)}(28,22) = q_4^{(4)}(29,24) = \mu_{10} \end{aligned}$$

in rows corresponding to states that have $p_{44} = 1$ for which transition t_{34} in Figure 3.11 is enabled. These nonzeros represent message departures from submodel 4 and therefore decrease the number of messages in it by 1.

The next group has the 17 nonzeros

$$\begin{aligned} & q_4^{(4)}(1,2) = q_4^{(4)}(3,4) = q_4^{(4)}(5,6) = q_4^{(4)}(7,8) \\ = \; & q_4^{(4)}(7,9) = q_4^{(4)}(8,10) = q_4^{(4)}(9,10) = q_4^{(4)}(11,12) \\ = \; & q_4^{(4)}(14,15) = q_4^{(4)}(14,16) = q_4^{(4)}(15,17) = q_4^{(4)}(16,17) \\ = \; & q_4^{(4)}(18,19) = q_4^{(4)}(19,20) = q_4^{(4)}(22,23) = q_4^{(4)}(24,25) \\ = \; & q_4^{(4)}(25,26) = \mu_1 \end{aligned}$$

corresponding to transitions t_{30} and t_{33} in Figure 3.11 that have the same rate. The former of these transitions take place when $p_{37} = p_{40} = 1$, whereas the latter takes place when $p_{42} = p_{45} = 1$. The nonzeros corresponding to transitions t_{30} and t_{33} represent the passing of a message to the session layer and the user, respectively.

The last group has the 12 nonzeros

$$\begin{aligned} & q_4^{(4)}(2,3) = \quad q_4^{(4)}(6,7) = \quad q_4^{(4)}(9,11) = \quad q_4^{(4)}(10,12) \\ = \; & q_4^{(4)}(13,14) = \quad q_4^{(4)}(16,18) = \quad q_4^{(4)}(17,19) = \quad q_4^{(4)}(20,21) \\ = \; & q_4^{(4)}(22,24) = \quad q_4^{(4)}(23,25) = \quad q_4^{(4)}(26,27) = \quad q_4^{(4)}(28,29) = \quad \mu_2 \end{aligned}$$

in rows corresponding to states that have $p_{39} = 1$ for which transition t_{31} in Figure 3.11 is enabled. These nonzeros represent the processing of a message at the session layer.

Regarding transition 5, we have the matrices

$$Q_5^{(1)} = \mathbf{e}_3\mathbf{e}_0^T + \mathbf{e}_6\mathbf{e}_1^T + \mathbf{e}_7\mathbf{e}_2^T + \mathbf{e}_8\mathbf{e}_3^T + \mathbf{e}_{10}\mathbf{e}_4^T + \mathbf{e}_{11}\mathbf{e}_5^T + \mathbf{e}_{12}\mathbf{e}_6^T + \mathbf{e}_{13}\mathbf{e}_9^T + \mathbf{e}_{14}\mathbf{e}_{10}^T,$$

$$Q_5^{(2)} = \mathbf{e}_0\mathbf{e}_1^T + \mathbf{e}_8\mathbf{e}_{10}^T + \mathbf{e}_{14}\mathbf{e}_{15}^T.$$

Since submodels 3 and 4 do not contribute to transition 2, we have $Q_5^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 3, 4$.

As for transitions 6, 7, and 8, we have the matrices

$$Q_6^{(2)} = \mathbf{e}_8\mathbf{e}_{14}^T + \mathbf{e}_{10}\mathbf{e}_{15}^T + \mathbf{e}_{12}\mathbf{e}_{13}^T, \quad Q_6^{(3)} = \mathbf{e}_0\mathbf{e}_4^T$$

since a last packet can depart from submodel 2 in states 8, 10, and 12, respectively, taking it into states 14, 15, and 13 and causing an empty submodel 3 to move to state 4,

$$Q_7^{(2)} = \mathbf{e}_4\mathbf{e}_{13}^T, \quad Q_7^{(3)} = \mathbf{e}_0\mathbf{e}_1^T$$

since a first packet can depart from submodel 2 in state 4 taking it into state 13 and causing an empty submodel 3 to move to state 1, and

$$Q_8^{(2)} = \mathbf{e}_{13}\mathbf{e}_5^T + \mathbf{e}_{14}\mathbf{e}_9^T + \mathbf{e}_{15}\mathbf{e}_{11}^T, \quad Q_8^{(3)} = \mathbf{e}_3\mathbf{e}_0^T$$

since an acknowledgment packet can depart from submodel 3 in state 3 taking it into state 0 and causing submodel 2 to move to states 5, 9, and 11, respectively, from states 13, 14, and 15. Submodels 1 and 4 do not contribute to transitions 6, 7, and 8. Therefore, we have $Q_k^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 1, 4$ and $k = 6, 7, 8$.

Regarding transition 9, we have the matrices

$$Q_9^{(3)} = \mathbf{e}_5\mathbf{e}_2^T,$$

$$
\begin{aligned}
Q_9^{(4)} = {} & \mathbf{e}_0\mathbf{e}_1^T + \mathbf{e}_1\mathbf{e}_5^T + \mathbf{e}_2\mathbf{e}_6^T + \mathbf{e}_3\mathbf{e}_7^T + \mathbf{e}_4\mathbf{e}_8^T + \mathbf{e}_6\mathbf{e}_{13}^T + \mathbf{e}_7\mathbf{e}_{14}^T + \mathbf{e}_8\mathbf{e}_{15}^T + \mathbf{e}_9\mathbf{e}_{16}^T \\
& + \mathbf{e}_{10}\mathbf{e}_{17}^T + \mathbf{e}_{11}\mathbf{e}_{18}^T + \mathbf{e}_{12}\mathbf{e}_{19}^T + \mathbf{e}_{16}\mathbf{e}_{22}^T + \mathbf{e}_{17}\mathbf{e}_{23}^T + \mathbf{e}_{18}\mathbf{e}_{24}^T + \mathbf{e}_{19}\mathbf{e}_{25}^T \\
& + \mathbf{e}_{20}\mathbf{e}_{26}^T + \mathbf{e}_{21}\mathbf{e}_{27}^T + \mathbf{e}_{26}\mathbf{e}_{28}^T + \mathbf{e}_{27}\mathbf{e}_{29}^T,
\end{aligned}
$$

and $Q_9^{(h)} = I_{|\mathcal{S}^{(h)}|}$ for $h = 1, 2$.

Finally, rates of transitions 1 through 9 are given by

$$
(\alpha_1, \ldots, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9) = (1, \ldots, 1, \mu_1, \mu_4, \mu_4, \mu_4, \mu_9) \, .
$$

As in (2.10), the communications protocol model defines the (4×4) block generator matrix

$$
Q = \begin{pmatrix}
Q(0,0) & Q(0,1) & Q(0,2) & Q(0,3) \\
 & Q(1,1) & Q(1,2) & \\
Q(2,0) & & Q(2,2) & \\
Q(3,0) & & Q(3,2) & Q(3,3)
\end{pmatrix},
$$

where the nonzero blocks are given from Table 3.1 and the matrix in (3.2) by

$$
\begin{aligned}
Q(0,0) = {} & \alpha_1 Q_1^{(1)} \otimes I_{13} \otimes I_1 \otimes I_{30} + \alpha_2 I_{15} \otimes Q_2^{(2)}(0:12, 0:12) \otimes I_1 \otimes I_{30} \\
& + \alpha_3 I_{15} \otimes I_{13} \otimes Q_3^{(3)}(0:0, 0:0) \otimes I_{30} + \alpha_4 I_{15} \otimes I_{13} \otimes I_1 \otimes Q_4^{(4)} \\
& + \alpha_5 Q_5^{(1)} \otimes Q_5^{(2)}(0:12, 0:12) \otimes I_1 \otimes I_{30} + Q_D(0,0), \\
Q(0,1) = {} & \alpha_7 I_{15} \otimes Q_7^{(2)}(0:12, 13:13) \otimes Q_7^{(3)}(0:0, 1:1) \otimes I_{30}, \\
Q(0,2) = {} & \alpha_6 I_{15} \otimes Q_6^{(2)}(0:12, 13:13) \otimes Q_6^{(3)}(0:0, 2:5) \otimes I_{30}, \\
Q(0,3) = {} & \alpha_6 I_{15} \otimes Q_6^{(2)}(0:12, 14:15) \otimes Q_6^{(3)}(0:0, 2:5) \otimes I_{30}, \\
Q(1,1) = {} & \alpha_1 Q_1^{(1)} \otimes I_1 \otimes I_1 \otimes I_{30} + \alpha_2 I_{15} \otimes Q_2^{(2)}(13:13, 13:13) \otimes I_1 \otimes I_{30} \\
& + \alpha_3 I_{15} \otimes I_1 \otimes Q_3^{(3)}(1:1, 1:1) \otimes I_{30} + \alpha_4 I_{15} \otimes I_1 \otimes I_1 \otimes Q_4^{(4)} \\
& + Q_D(1,1), \\
Q(1,2) = {} & \alpha_3 I_{15} \otimes I_1 \otimes Q_3^{(3)}(1:1, 2:5) \otimes I_{30}, \\
Q(2,0) = {} & \alpha_8 I_{15} \otimes Q_8^{(2)}(13:13, 0:12) \otimes Q_8^{(3)}(2:5, 0:0) \otimes I_{30},
\end{aligned}
$$

$$
\begin{aligned}
Q(2,2) &= \alpha_1 Q_1^{(1)} \otimes I_1 \otimes I_4 \otimes I_{30} + \alpha_2 I_{15} \otimes Q_2^{(2)}(13:13,13:13) \otimes I_4 \otimes I_{30} \\
&\quad + \alpha_3 I_{15} \otimes I_1 \otimes Q_3^{(3)}(2:5,2:5) \otimes I_{30} + \alpha_4 I_{15} \otimes I_1 \otimes I_4 \otimes Q_4^{(4)} \\
&\quad + \alpha_9 I_{15} \otimes I_1 \otimes Q_9^{(3)}(2:5,2:5) \otimes Q_9^{(4)} + Q_D(2,2), \\
Q(3,0) &= \alpha_8 I_{15} \otimes Q_8^{(2)}(14:15,0:12) \otimes Q_8^{(3)}(2:5,0:0) \otimes I_{30}, \\
Q(3,2) &= \alpha_2 I_{15} \otimes Q_2^{(2)}(14:15,13:13) \otimes I_4 \otimes I_{30}, \\
Q(3,3) &= \alpha_1 Q_1^{(1)} \otimes I_2 \otimes I_4 \otimes I_{30} + \alpha_2 I_{15} \otimes Q_2^{(2)}(14:15,14:15) \otimes I_4 \otimes I_{30} \\
&\quad + \alpha_3 I_{15} \otimes I_2 \otimes Q_3^{(3)}(2:5,2:5) \otimes I_{30} + \alpha_4 I_{15} \otimes I_2 \otimes I_4 \otimes Q_4^{(4)} \\
&\quad + \alpha_5 Q_5^{(1)} \otimes Q_5^{(2)}(14:15,14:15) \otimes I_4 \otimes I_{30} \\
&\quad + \alpha_9 I_{15} \otimes I_2 \otimes Q_9^{(3)}(2:5,2:5) \otimes Q_9^{(4)} + Q_D(3,3).
\end{aligned}
$$

Each diagonal block $Q_D(p,p)$ for $p = 0,\ldots,3$ is defined as before and will not be given explicitly.

The number of floating-point values to be stored in this representation is 9 for the rates, $20 + 12 + 3 + 44 = 79$ for the local transition matrices, and $(9+3) + (3+1) + (1+1) + (3+1) + (1+20) = 43$ for the synchronizing transition matrices, thus altogether, 131. When $C = 4$, we have $N = 13$, and this number becomes 2,342 [49]. On the other hand, Q is of order 11,700, and the number of nonzeros in its off-diagonal part is 48,330 for $C = 1$, whereas Q is of order 1,632,600 and has a total of 9,732,330 nonzero off-diagonal entries [49] when $C = 4$ for which $|\mathcal{S}^{(2)}| = 546$ and $|\mathcal{S}^{(3)}| = 229$.

Whether a minimum number of reachable state space partitions, $N_{\min}$, is computed through the approach devised in [49] is what we investigate now. A quick look into the four reachable state space partitions of the communications protocol model in Table 3.1 reveals that it is possible to merge $\mathcal{R}_1$ and $\mathcal{R}_2$ or $\mathcal{R}_2$ and $\mathcal{R}_3$ to obtain $N_{\min} = 3$ as the minimum number of reachable state space partitions. In fact, the merge-based and the refinement-based algorithms introduced at the beginning of this chapter both yield three partitions given reachable state space $\mathcal{R}$. A similar observation is made for the reachable state space partitions of the availability model in Table 2.1. Therein, it is possible to merge $\mathcal{R}_0$ with one of the three reachable state space partitions $\mathcal{R}_1$, $\mathcal{R}_2$, or $\mathcal{R}_3$ and again obtain $N_{\min} = 3$ as the minimum. However, this is not the case for the reachable state space partitions of the polling model in Table 2.2 where we already have $N_{\min} = 6$ as the minimum.

When reachable state space partitions can be merged as in the availability and communications protocol models, it is done at the expense of relaxing the constraint in (3.1) regarding nonintersecting subsets of submodel state spaces. That is, the constraint $\mathcal{S}_p^{(h)} \cap \mathcal{S}_w^{(h)} = \varnothing$, $p \neq w$ for $p, w = 0,\ldots,N_h - 1$ is relaxed so that we only have $\mathcal{S}_p^{(h)} \subseteq \mathcal{S}^{(h)}$ for dimension h, where $h = 1,\ldots,H$.

Let us walk through the first alternative in the communications protocol model which expresses $\mathcal{R}$ using $N_{\min} = 3$ partitions as in Table 3.2. We have renumbered the reachable state space partitions of $\mathcal{R}$ so that $\mathcal{R}_0$ is $\mathcal{R}_{\tilde{0}}$,

$\mathcal{R}_1 \cup \mathcal{R}_2$ is $\mathcal{R}_{\tilde{1}}$, and $\mathcal{R}_3$ is $\mathcal{R}_{\tilde{2}}$. Hence,

$$\mathcal{R} = \bigcup_{p=\tilde{0}}^{\tilde{2}} \mathcal{R}_p \quad \text{with} \quad \mathcal{R}_p = \bigtimes_{h=1}^{4} \mathcal{R}_p^{(h)} \quad \text{for } p = \tilde{0}, \tilde{1}, \tilde{2},$$

where

$$\mathcal{R}_{\tilde{0}} = \mathcal{S}_0^{(1)} \times \mathcal{S}_0^{(2)} \times \mathcal{S}_0^{(3)} \times \mathcal{S}_0^{(4)}\,, \quad \mathcal{R}_{\tilde{1}} = \mathcal{S}_0^{(1)} \times \mathcal{S}_1^{(2)} \times (\mathcal{S}_1^{(3)} \cup \mathcal{S}_2^{(3)}) \times \mathcal{S}_0^{(4)}\,,$$

$$\mathcal{R}_{\tilde{2}} = \mathcal{S}_0^{(1)} \times \mathcal{S}_2^{(2)} \times \mathcal{S}_2^{(3)} \times \mathcal{S}_0^{(4)}\,.$$

Note the union of $\mathcal{S}_1^{(3)}$ and $\mathcal{S}_2^{(3)}$ in the expression for $\mathcal{R}_{\tilde{1}}$. Since $\mathcal{S}_2^{(3)}$ also appears in $\mathcal{R}_{\tilde{2}}$, clearly the constraint in (3.1) regarding nonintersecting subsets of submodel state spaces no longer holds for submodel 3.

Table 3.2 Alternative mapping between reachable state space partitions and submodel states in communications protocol model

Partition	Submodel 1	Submodel 2	Submodel 3	Submodel 4	# of states
$\mathcal{R}_{\tilde{0}}$	0:14	0:12	0:0	0:29	15 . 13 . 1 . 30 = 5,850
$\mathcal{R}_{\tilde{1}}$	0:14	13:13	1:5	0:29	15 . 1 . 5 . 30 = 2,250
$\mathcal{R}_{\tilde{2}}$	0:14	14:15	2:5	0:29	15 . 2 . 4 . 30 = 3,600

The (3×3) interaction matrix which captures the transitions is then

$$\begin{array}{c} \\ \mathcal{R}_{\tilde{0}} \\ \mathcal{R}_{\tilde{1}} \\ \mathcal{R}_{\tilde{2}} \end{array} \begin{array}{c} \begin{array}{ccc} \mathcal{R}_{\tilde{0}} & \mathcal{R}_{\tilde{1}} & \mathcal{R}_{\tilde{2}} \end{array} \\ \begin{pmatrix} \{1,2,3,4,5\} & \{6,7\} & \{6\} \\ \{8\} & \{1,2,3,4,9\} & \\ \{8\} & \{2\} & \{1,2,3,4,5,9\} \end{pmatrix} \end{array}. \tag{3.3}$$

As in (2.10), the communications protocol model alternatively defines the (3×3) block generator matrix

$$Q = \begin{pmatrix} Q(\tilde{0},\tilde{0}) & Q(\tilde{0},\tilde{1}) & Q(\tilde{0},\tilde{2}) \\ Q(\tilde{1},\tilde{0}) & Q(\tilde{1},\tilde{1}) & \\ Q(\tilde{2},\tilde{0}) & Q(\tilde{2},\tilde{1}) & Q(\tilde{2},\tilde{2}) \end{pmatrix},$$

where the nonzero blocks are given from Table 3.2 and the matrix in (3.3) by

$$\begin{aligned} Q(\tilde{0},\tilde{0}) &= Q(0,0), \\ Q(\tilde{0},\tilde{1}) &= \alpha_6 I_{15} \otimes Q_6^{(2)}(0:12,13:13) \otimes Q_6^{(3)}(0:0,1:5) \otimes I_{30} \\ &\quad + \alpha_7 I_{15} \otimes Q_7^{(2)}(0:12,13:13) \otimes Q_7^{(3)}(0:0,1:5) \otimes I_{30}, \\ Q(\tilde{0},\tilde{2}) &= Q(0,3), \\ Q(\tilde{1},\tilde{0}) &= \alpha_8 I_{15} \otimes Q_8^{(2)}(13:13,0:12) \otimes Q_8^{(3)}(1:5,0:0) \otimes I_{30}, \end{aligned}$$

$$
\begin{aligned}
Q(\tilde{1},\tilde{1}) &= \alpha_1 Q_1^{(1)} \otimes I_1 \otimes I_5 \otimes I_{30} + \alpha_2 I_{15} \otimes Q_2^{(2)}(13:13,13:13) \otimes I_5 \otimes I_{30} \\
&\quad + \alpha_3 I_{15} \otimes I_1 \otimes Q_3^{(3)}(1:5,1:5) \otimes I_{30} + \alpha_4 I_{15} \otimes I_1 \otimes I_5 \otimes Q_4^{(4)} \\
&\quad + \alpha_9 I_{15} \otimes I_1 \otimes Q_9^{(3)}(1:5,1:5) \otimes Q_9^{(4)} + Q_D(\tilde{1},\tilde{1}), \\
Q(\tilde{2},\tilde{0}) &= Q(3,0), \\
Q(\tilde{2},\tilde{1}) &= \alpha_2 I_{15} \otimes Q_2^{(2)}(14:15,13:13) \otimes (0_{4\times 1}\ I_4) \otimes I_{30}, \\
Q(\tilde{2},\tilde{2}) &= Q(3,3),
\end{aligned}
$$

where $0_{4\times 1}$ is a (4×1) zero matrix (i.e., a 4-vector of 0s) that pads the identity matrix of order 4 to the left with an empty column so that we have a (4×5) matrix as the third Kronecker product factor in $Q(\tilde{2},\tilde{1})$. Diagonal block $Q_D(\tilde{1},\tilde{1})$ is defined as before and will not be given explicitly. Note that with this alternative reachable state space partitioning, neither the number of nonzero entries to be stored with the Kronecker representation nor the structure of its underlying generator matrix, Q, changes.

A smaller number of reachable state space partitions, N, implies larger block sizes in the nested partitioning of Q. For a particular level in the nested partitioning, this translates to less index manipulations, access to a larger number of consecutive memory locations, and larger blocks along the diagonal that can be used in preconditioned iterative methods with the Kronecker representation. Future work may consider investigating the merits of using a smaller value of N as discussed here to find out whether it brings any improvement in analyses.

3.4 Specification of Kronecker-Structured CTMCs

There are different ways in which information required to process the Kronecker representation of a CTMC during analysis can be specified on a computer. In this section, we describe one such specification that is based on the file format in the `NSolve` package of the APNN toolbox [7, 22] and can be used with the representation in (2.10).

For a Kronecker-structured multidimensional model with H submodels, $H + 2$ files are used. Files belonging to a particular model specification are given a common name and are distinguished by appending indices followed by a dot and a three letter suffix. The file with suffix `.spa` describes the mapping between reachable state space partitions and submodel state space partitions. The other files follow the model name with the indices `0` through `H` and are given the suffix `.mat`. The file with index `0` specifies the transitions among reachable state space partitions and their rates. The file with index `h` specifies the state space partitions and transition matrices associated with submodel h for $h = 1, \ldots, H$.

We demonstrate the file format on the communications protocol model that has multiple reachable state space partitions and give `courier` as the common name to files associated with its specification. In the communications protocol model, the specification files are therefore named `courier.spa`, `courier0.mat`, `courier1.mat`, ..., `courierH.mat`.

File `courier.spa` corresponding to Table 3.1 is given by

```
0 0 0 0
0 1 1 0
0 1 2 0
0 2 2 0
```

This file has as many rows as there are reachable state space partitions, N. On each row, there are H integer entries separated by white space characters. Entry h of row $p+1$ corresponds to the index of the partition of submodel state space $\mathcal{S}^{(h)}$ mapped to reachable state space partition $\mathcal{R}_p$ for $p = 0, \ldots, N-1$ and $h = 1, \ldots, H$.

File `courier0.mat` corresponding to the $(N \times N)$ interaction matrix in (3.2) is given by

```
1
0
4
8
0 1 1.000000000000000e+00
0 2 1.000000000000000e+00
0 3 1.000000000000000e+00
0 4 1.000000000000000e+00
0 5 8.771929824561405e+03
1 7 4.821600771456124e+02
2 6 4.821600771456124e+02
3 6 4.821600771456124e+02
5
1 1 1.000000000000000e+00
1 2 1.000000000000000e+00
1 3 1.000000000000000e+00
1 4 1.000000000000000e+00
2 3 1.000000000000000e+00
6
0 8 4.821600771456124e+02
2 1 1.000000000000000+e00
2 2 1.000000000000000+e00
2 3 1.000000000000000+e00
2 4 1.000000000000000+e00
2 9 1.449275362318841e+03
8
0 8 4.821600771456124e+02
```

```
2 2 1.000000000000000e+00
3 1 1.000000000000000e+00
3 2 1.000000000000000e+00
3 3 1.000000000000000e+00
3 4 1.000000000000000e+00
3 5 8.771929824561405e+03
3 9 1.449275362318841e+03
```

The integer in the first line of this file stands for the number of matrices defined in the file. The integers in the second and third lines of the file stand, respectively, for the starting index of reachable state space partitions and number of reachable state space partitions, N. It is the understanding that block rows and block columns of Q are numbered consecutively starting from 0 up to $N-1$. Then starting with row index 0, the file provides consecutively each row of the interaction matrix in what is called the coefficient sparse format for matrices. The first integer corresponding to each row tells how many nonzeros exist in that row. This integer is followed by that many rows each of which has three entries separated by white space characters. The first entry is an integer indicating the column index associated with the transition, the second entry which is also an integer indicates the transition number, and the third entry which is a real number in scientific notation with 16 decimal digits for the mantissa and 2 decimal digits for the exponent indicates the rate of the transition. Since rows appear in increasing index, there is no need to specify the row index as long as one knows the starting index of rows and how many rows there are. Note that if the interaction matrix has nz_{im} nonzeros, the file will have $3 + N + nz_{im}$ lines, which is 34 for the communications protocol model since $N = 4$ and $nz_{im} = 27$.

Each of the remaining H files is associated with a different submodel and specifies its state space partitions and transition matrices. File `courier1.mat` corresponding to submodel 1 of the communications protocol model is given by

```
2
1
0
15

1
1
1 7.352941176470588e+03
2
2 8.771929824561405e+03
4 7.352941176470588e+03
2
3 1.006036217303823e+03
5 7.352941176470588e+03
```

```
1
6 7.352941176470588e+03
1
5 8.771929824561405e+03
2
6 1.006036217303823e+03
9 7.352941176470588e+03
2
7 8.771929824561405e+03
10 7.352941176470588e+03
2
8 1.006036217303823e+03
11 7.352941176470588e+03
1
12 7.352941176470588e+03
1
10 1.006036217303823e+03
1
11 8.771929824561405e+03
2
12 1.006036217303823e+03
13 7.352941176470588e+03
1
14 7.352941176470588e+03
1
14 1.006036217303823e+03
0

5
0
0
0
1
0 1.000000000000000e+00
0
0
1
1 1.000000000000000e+00
1
2 1.000000000000000e+00
1
3 1.000000000000000e+00
0
1
4 1.000000000000000e+00
```

```
1
5 1.000000000000000e+00
1
6 1.000000000000000e+00
1
9 1.000000000000000e+00
1
10 1.000000000000000e+00
```

The integer in the first line of the file for submodel h gives its number of transition matrices that are different than identity. The integer in the second line gives the number of partitions, N_h, of its state space $\mathcal{S}^{(h)}$. Then starting from partition 0, the integers on lines 3 through $2 + N_h$, one on each line, provide the starting index of partition p of submodel h, that is, $\mathcal{S}_p^{(h)}$, for $p = 0, \ldots, N_h - 1$. The integer on line $3 + N_h$ gives the submodel state space size $|\mathcal{S}^{(h)}|$. The first $N_h + 3$ lines are followed after a blank line by transition matrices of submodel h. Each transition matrix is separated from the next by an empty line for readability. The description of each transition matrix starts with an integer providing the index of the transition on a separate line. Then the particular transition matrix is given row by row starting with index 0 up to $|\mathcal{S}^{(h)}| - 1$ consecutively. Each row specification starts by the number of nonzeros in that row in a separate line which is followed by that many lines, each line providing the integer column index and its corresponding real nonzero value again in scientific notation. It is important to have a 0 for each row that does not have any nonzeros in the file since row indices are not stored. Note that if non-identity transition matrix $Q_k^{(h)}$ has $nz_{Q_k^{(h)}}$ nonzeros, the corresponding file will have $1 + |\mathcal{S}^{(h)}| + nz_{Q_k^{(h)}}$ lines for that matrix. Thus, the file for submodel h will have altogether

$$3 + N_h + \sum_{k \in \mathcal{K}^{(h)}} (2 + |\mathcal{S}^{(h)}| + nz_{Q_k^{(h)}})$$

lines, where

$$\mathcal{K}^{(h)} = \{k \mid k \in \{1, \ldots, K\} \text{ and } Q_k^{(h)} \neq I\},$$

that is, the set of transitions in which submodel h participates. Now, recall that $\mathcal{S}^{(1)} = 15$, $N_1 = 1$, $\mathcal{K}^{(1)} = \{1, 5\}$, $nz_{Q_1^{(1)}} = 20$, and $nz_{Q_5^{(1)}} = 9$, implying `courier1.mat` must be 67 lines.

Submodel 2 of the communications protocol model participates in five transitions with $\mathcal{K}^{(2)} = \{2, 5, 6, 7, 8\}$, and its state space $\mathcal{S}^{(2)}$ of $|\mathcal{S}^{(2)}| = 16$ states is partitioned into $N_2 = 3$ subsets. Since $nz_{Q_2^{(2)}} = 12$, $nz_{Q_5^{(2)}} = 3$, $nz_{Q_6^{(2)}} = 3$, $nz_{Q_7^{(2)}} = 1$, and $nz_{Q_8^{(2)}} = 3$, file `courier2.mat` corresponding to submodel 2 is given by the 118 lines:

```
5
3
```

```
0
13
14
16

2
0
2
2 8.680555555555555e+02
6 8.680555555555555e+02
1
3 1.165501165501165e+03
1
4 1.282051282051282e+04
0
2
2 2.293577981651376e+03
6 2.293577981651376e+03
1
7 1.165501165501165e+03
1
8 1.282051282051282e+04
0
1
0 4.587155963302752e+03
1
12 1.736111111111111e+03
1
5 1.736111111111111e+03
0
0
0
1
13 1.736111111111111e+03

5
1
1 1.000000000000000e+00
0
0
0
0
0
0
0
```

```
1
10 1.000000000000000e+00
0
0
0
0
0
1
15 1.000000000000000e+00
0

6
0
0
0
0
0
0
0
0
1
14 1.000000000000000e+00
0
1
15 1.000000000000000e+00
0
1
13 1.000000000000000e+00
0
0
0

7
0
0
0
0
1
13 1.000000000000000e+00
0
0
0
0
0
0
```

```
0
0
0
0
0

8
0
0
0
0
0
0
0
0
0
0
0
0
0
1
5 1.000000000000000e+00
1
9 1.000000000000000e+00
1
11 1.000000000000000e+00
```

Submodel 3 of the communications protocol model participates in five transitions with $\mathcal{K}^{(3)} = \{3, 6, 7, 8, 9\}$, and its state space $\mathcal{S}^{(3)}$ of $|\mathcal{S}^{(3)}| = 6$ states is partitioned into $N_3 = 3$ subsets. Since $nz_{Q_3^{(3)}} = 3$ and $nz_{Q_6^{(3)}} = nz_{Q_7^{(3)}} = nz_{Q_8^{(1)}} = nz_{Q_9^{(3)}} = 1$, file `courier3.mat` corresponding to submodel 3 is given by the 53 lines:

```
5
3
0
1
2
6

3
0
1
2 1.165501165501165e+03
1
3 4.587155963302752e+03
```

```
0
1
5 1.165501165501165e+03
0

6
1
4 1.000000000000000e+00
0
0
0
0
0

7
1
1 1.000000000000000e+00
0
0
0
0
0

8
0
0
0
1
0 1.000000000000000e+00
0
0

9
0
0
0
0
0
1
2 1.000000000000000e+00
```

Submodel 4 of the communications protocol model participates in two transitions with $\mathcal{K}^{(4)} = \{4, 9\}$, and its state space $\mathcal{S}^{(4)}$ of $|\mathcal{S}^{(4)}| = 30$ states is partitioned into $N_4 = 1$ subset. Since $nz_{Q_4^{(4)}} = 44$ and $nz_{Q_9^{(4)}} = 20$, implying file `courier4.mat` corresponding to submodel 4 is given by the 132 lines:

```
2
1
0
30

4
0
1
2 8.771929824561405e+03
1
3 1.006036217303823e+03
1
4 8.771929824561405e+03
1
0 4.000000000000000e+03
1
6 8.771929824561405e+03
1
7 1.006036217303823e+03
2
8 8.771929824561405e+03
9 8.771929824561405e+03
2
1 4.000000000000000e+03
10 8.771929824561405e+03
2
10 8.771929824561405e+03
11 1.006036217303823e+03
2
2 4.000000000000000e+03
12 1.006036217303823e+03
1
12 8.771929824561405e+03
1
3 4.000000000000000e+03
1
14 1.006036217303823e+03
2
15 8.771929824561405e+03
16 8.771929824561405e+03
2
5 4.000000000000000e+03
17 8.771929824561405e+03
2
17 8.771929824561405e+03
```

```
18 1.006036217303823e+03
2
6 4.000000000000000e+03
19 1.006036217303823e+03
1
19 8.771929824561405e+03
2
7 4.000000000000000e+03
20 8.771929824561405e+03
2
9 4.000000000000000e+03
21 1.006036217303823e+03
1
11 4.000000000000000e+03
2
23 8.771929824561405e+03
24 1.006036217303823e+03
2
13 4.000000000000000e+03
25 1.006036217303823e+03
1
25 8.771929824561405e+03
2
14 4.000000000000000e+03
26 8.771929824561405e+03
2
16 4.000000000000000e+03
27 1.006036217303823e+03
1
18 4.000000000000000e+03
2
22 4.000000000000000e+03
29 1.006036217303823e+03
1
24 4.000000000000000e+03

9
1
1 1.000000000000000e+00
1
5 1.000000000000000e+00
1
6 1.000000000000000e+00
1
7 1.000000000000000e+00
```

```
1
8 1.000000000000000e+00
0
1
13 1.000000000000000e+00
1
14 1.000000000000000e+00
1
15 1.000000000000000e+00
1
16 1.000000000000000e+00
1
17 1.000000000000000e+00
1
18 1.000000000000000e+00
1
19 1.000000000000000e+00
0
0
0
1
22 1.000000000000000e+00
1
23 1.000000000000000e+00
1
24 1.000000000000000e+00
1
25 1.000000000000000e+00
1
26 1.000000000000000e+00
1
27 1.000000000000000e+00
0
0
0
0
1
28 1.000000000000000e+00
1
29 1.000000000000000e+00
0
0
```

3.5 Comparison of Cartesian Product Partitioning Algorithms

The proposed merge- and refinement-based Cartesian product partitioning algorithms are coded in [113] and tested for the quality of their computed solutions and their scalability on multidimensional reachable state spaces. We are interested in seeing how the computed value of the number of reachable state space partitions, N, compares to the minimum, $N_{\min}$, possible and how time and space requirements of the algorithms change as the reachable state space, $\mathcal{R}$, becomes larger among different models.

In order to determine $N_{\min}$, we follow the approach discussed below. The states of each connected component in the graph that is obtained by removing the separators from the unit distance graph of $\mathcal{R}$ are in the same partition in the minimum Cartesian product partitioning of $\mathcal{R}$ from Lemma 4.1 in [111]. The proof of this result is lengthy and employs a number of contradictions [108]. Using this result, we conclude that each partition in the minimum Cartesian product partitioning of $\mathcal{R}$ is a union of partitions obtained by removing the separators from the unit distance graph of $\mathcal{R}$. In order to determine the number of partitions in the minimum Cartesian product partitioning of a model, $N_{\min}$, all possible Cartesian product partitionings satisfying this condition are computed and the minimum chosen. When there are a small number of separators and intersecting separator pairs, this is straightforward. However, the number of Cartesian product partitionings to compute increases exponentially with the number of separators and intersecting separator pairs. This hints at the difficulty inherent in computing the minimum Cartesian product partitioning of a multidimensional reachable state space.

Table 3.3 summarizes the properties of the models employed in the experiments. The integer parameters of these models are $S = 2$ and $C_h = 10$ with $h = 1, \ldots, H$ for polling, $C_h = 4$ with $h = 1, \ldots, H$ for production line, and $C \in \{1, 2, 3, 4\}$ for communications protocol. Results with some other models can be found in [108, 111]. Column nz reports the number of nonzeros in the off-diagonal part of Q, and column $nz_{\otimes}$ reports the number of floating-point values stored in its Kronecker-structured specification. The seventh column of the table titled $N_{\min}$ reports the number of partitions in the minimum Cartesian product partitioning of $\mathcal{R}$. The eighth column titled Sep and the ninth column titled $Sep\bigcap$ report the number of separators and the number of intersecting separator pairs (i.e., separator pairs including edges along different dimensions that are incident to the same vertex) in the unit distance graph of $\mathcal{R}$, respectively.

The reachable state space of a multidimensional model depends on the interaction of its submodels. Among the four different models considered in Table 3.3, the communications protocol model appears to be the more complicated one since its unit distance graph has separators and intersect-

Table 3.3 Properties of models

Model	H	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{R}\rvert$	nz	$nz_{\otimes}$	$N_{\min}$	Sep	$Sep^{\bigcap}$
Availability	3	1,728	1,728	8,010	99	1	0	0
	4	20,736	20,736	116,874	138	1	0	0
	5	248,832	248,832	1,651,338	180	1	0	0
	6	2,985,984	2,985,984	22,802,058	225	1	0	0
	7	35,831,808	35,831,808	309,456,522	273	1	0	0
	8	429,981,696	429,981,696	4,088,083,549	324	1	0	0
Polling	3	238,328	25,443	112,791	453	6	0	0
	4	14,776,336	479,886	2,594,108	604	10	0	0
	5	916,132,832	8,065,860	51,259,835	755	15	0	0
	6	56,800,235,584	125,839,395	917,647,302	906	21	0	0
	7	3,521,614,606,208	1,863,521,121	15,321,499,039	1,057	28	0	0
Production	4	5,625	5,625	8,500	83	1	0	0
	5	84,375	84,375	165,000	115	1	0	0
	6	1,265,625	1,265,625	3,037,500	147	1	0	0
	7	18,984,375	18,984,375	54,000,000	179	1	0	0
	8	284,765,625	284,765,625	936,562,500	211	1	0	0
Protocol $C = 1$	4	43,200	11,700	48,330	131	3	2	1
$C = 2$	4	838,350	84,600	410,160	290	5	4	2
$C = 3$	4	8,593,200	419,400	2,281,620	854	7	6	3
$C = 4$	4	56,265,300	1,632,600	9,732,330	2,342	9	8	4

ing separators. There are no separators in the other models since their unit distance graphs do not include any conflicting edges.

Table 3.4 Results of Cartesian product partitioning of reachable state space for models with lexicographical ordering of states

			Merge		Refinement		
Model	H	N	Time	Memory	N	Time	Memory
Availability	3	1	0	0	1	0	8
	4	1	0	8	1	0	16
	5	1	0	8	1	1	109
	6	1	6	8	1	16	1,512
	7	1	89	8	-	-	-
	8	1	1,308	8	-	-	-
Polling	3	6	0	8	6	0	15
	4	10	1	8	10	1	188
	5	15	16	8	15	34	3,395
	6	21	320	8	-	-	-
	7	28	5,791	8	-	-	-
Production	4	1	0	8	1	0	0
	5	1	0	8	1	0	42
	6	1	3	8	1	6	646
	7	1	48	8	1	156	10,259
	8	1	916	8	-	-	-
Protocol $C = 1$	4	3	0	8	3	0	12
$C = 2$	4	5	0	8	5	0	40
$C = 3$	4	7	1	8	7	1	165
$C = 4$	4	9	2	8	9	6	634

In Table 3.4, we present the results of experiments with the models in Table 3.3 when the states in $\mathcal{R}$ are processed in lexicographical order. If the states do not appear in this order, then they may be easily sorted into this order. We report the computation time rounded to the nearest second under column Time and the allocated space to carry out the computation in MB under column Memory. The refinement-based algorithm is not able to yield a solution in seven- and eight-dimensional availability models, six- and seven-dimensional polling models, and eight-dimensional production line model due to memory limitations. When states are processed in lexicographical order, merge-based and refinement-based algorithms both compute the optimal solution $N_{\min}$ for all models considered. We remark that this is not the case for some of the models in [108, 111] where the refinement-based algorithm almost always provides a winner in terms of quality of the solution. However, the merge-based algorithm requires less time and space than the refinement-based algorithm in all models as $|\mathcal{R}|$ becomes larger, suggesting its use in larger problems.

It is also possible to investigate the effects of processing the states in $\mathcal{R}$ in different orders by using the code in [113]. When the refinement-based algorithm processes the states in random order, the number of reachable state space partitions, N, it computes and its space requirement do not change, but its time requirement increases. The increase in time cannot be due to refinement of the graph and updating of the separators since Sep is either zero or small in the models considered, but it must be due to construction of the graph, suggesting more efficient utilization of the cache while processing states in lexicographical order. This must be because each state becomes a child of the last accessed state in the AVL tree, implying more efficient insertion to the tree when states are processed in lexicographical order.

The order of processing states in $\mathcal{R}$ turns out to be very influential on the performance of the merge-based algorithm. For instance, in the smallest communications protocol model of Table 3.3 which has 11,700 reachable states and $N_{\min} = 3$, the merge-based algorithm computes N values ranging from 2,925 to 3,127 when the states are processed in 51 different random orderings. These values of N are almost 1,000 times that of $N_{\min}$ in Table 3.4 computed by the same algorithm when the states are processed in lexicographical order. To understand this phenomenon, first recall that two partitions are merged along dimension h if their subsets of states are identical in all dimensions except h. Second, when the states are processed in lexicographical order, a partition ends up being merged with all possible partitions along a dimension before it is merged with a partition in a different dimension. Third, when the states are processed in random order, partitions are merged along random dimensions depending on the processing order of states, implying that all possible partitions along a particular dimension are not considered before switching to a different dimension. As a result of these, when states are processed in random order, N increases substantially for the merged-based algorithm. Since time and space requirements of the merge-based algorithm

depend on N, they also increase substantially which agrees with our earlier analysis. The time taken by the merge-based algorithm becomes comparable to the time taken by the refinement-based algorithm. However, its space requirement is still better than the space requirement of the refinement-based algorithm. See [108, 111] for a detailed study.

In conclusion, although it may be more time- and space-consuming, the refinement-based algorithm almost always computes partitionings with a smaller value of N than the merge-based algorithm. In many cases, the partitionings computed by the refinement-based algorithm yield $N_{\min}$. Furthermore, the refinement-based algorithm is insensitive to the processing order of states in $\mathcal{R}$. However, time and space requirements of the refinement-based algorithm do not scale very well with $|\mathcal{R}|$. For large models, we therefore recommend the merge-based algorithm and processing the states in lexicographical order.

Chapter 4
Preprocessing

In this chapter, we discuss preprocessing techniques to aid the analysis of MCs based on Kronecker products and improve time and memory requirements. There are a number of techniques that can be used to put the Kronecker representation into a more favorable form before solvers take over. These are reordering, grouping, lumping, and analyzing diagonal blocks for common Schur factors. Furthermore, when the reachable state space of the underlying MC is countably infinite, it needs to be truncated judiciously to enable analysis. We discuss them in order.

4.1 Reordering and Grouping

We assume that Q represents the generator matrix of a CTMC in block Kronecker form with multiple reachable state space partitions as in (2.10). The block form of Q in (2.10) is preserved up to a symmetric permutation, that is, up to a reordering of the reachable state space, $\mathcal{R}$, in which the reachable state space partitions $\mathcal{R}_p$ for $p = 0, \ldots, N-1$, the states in submodel state spaces $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$, or the submodels themselves are renumbered. Yet, there may be multiple ways in which the number of Kronecker product terms $|\mathcal{K}_{p,w}|$ in block (p, w) for $p, w = 0, \ldots, N-1$ and the number of factors in each Kronecker product, H, are chosen. Obviously, the choice $(|\mathcal{K}|, H) = (1, 1)$ suggests a flat representation and is assumed to be impossible due to memory limitations. As H decreases toward 1, the Kronecker representation becomes flatter, implying increased storage requirements. On the other hand, if $|\mathcal{K}|$ were 1, then Q could be analyzed along each dimension independently. Hence, $|\mathcal{K}|$ is normally assumed to be larger than 1. Observe

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_4

that it would be advantageous to be able to make $|\mathcal{K}|$ as small as possible without changing H, since then we would be decreasing the number of terms in the Kronecker representation of Q and making the matrices $Q_k^{(h)}$ denser.

Reordering in MCs based on Kronecker products refers to renumbering reachable state space partitions, states in submodel state spaces, or submodels. We remark that the first kind of reordering corresponds to a symmetric permutation of the interaction matrix in (2.11), and the second kind of reordering corresponds to a symmetric permutation of the transition matrices $Q_k^{(h)}$ associated with the renumbered state space $\mathcal{S}^{(h)}$. As indicated in [26, 137], reordering of the third kind may be used to reduce the overhead associated with vector–Kronecker product multiplication in the presence of functional transitions. Furthermore, reordering of all kinds can change the nonzero structure of Q and thereby have an effect on the convergence of iterative methods sensitive to its nonzero structure [96]. Hence, by the help of reordering, it may be possible to symmetrically permute the nonzero structure of Q to a more favorable form for the iterative method of choice. In doing this, the nonzero structure of the interaction matrix can be used to arrange the block nonzero structure of Q and the nonzero structure of $\sum_{k\in\mathcal{K}_{p,w}} Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$ to arrange the contribution of factor h for $h = 1, \ldots, H$ to the nonzero structure of block $Q(p, w)$ for $p, w = 0, \ldots, N-1$.

In the following, we show how one can take advantage of reordering of the second and third kinds so as to reduce the number of nonzeros in the LU factors of diagonal blocks while controlling the order of diagonal blocks in the nested block partitioning of Q.

Example 5. Table 4.1 provides the four nested block partitionings along the diagonal of Q induced by the Kronecker structure of the communications protocol model with $C = 1$ in Chapter 3. Columns b_l and o_l, respectively,

Table 4.1 Four nested block partitionings along the diagonal of Q in communications protocol model with $C = 1$

	Level 0		Level 1			Level 2			Level 3		
Partition	b_0	o_0	b_1	c_1	o_1	b_2	c_2	o_2	b_3	c_3	o_3
0	1	5,850	15	6	390	195	195	30	195	195	30
1	1	450	15	15	30	15	15	30	15	15	30
2	1	1,800	15	15	120	15	15	120	60	45	30
3	1	3,600	15	6	240	30	30	120	120	90	30
$\Sigma =$	4		60	42		255	255		390	345	

list the number and the order of diagonal blocks in each reachable state space partition for $l = 0, 1, 2, 3$ as in Chapter 2. We provide some more information under columns c_l (which denotes the number of candidate blocks) for $l = 1, 2, 3$ which is to be discussed in Section 4.3. Since $N > 1$, there

exists a partitioning at level 0 with diagonal blocks $Q(0,0)$, $Q(1,1)$, $Q(2,2)$, and $Q(3,3)$. These diagonal blocks can be partitioned further as defined by submodel 1 at level 1, by submodels 1 and 2 at level 2, and by submodels 1 through 3 at level 3. At level 4, each diagonal block is a scalar and therefore of no interest. We provide the same information in Tables 4.2, 4.3, and 4.4 for the

Table 4.2 Four nested block partitionings along the diagonal of Q in communications protocol model with $C = 2$

	Level 0		Level 1			Level 2			Level 3		
Partition	b_0	o_0	b_1	c_1	o_1	b_2	c_2	o_2	b_3	c_3	o_3
0	1	22,500	15	6	1,500	750	750	30	750	750	30
1	1	18,000	15	6	1,200	30	30	600	600	450	30
2	1	450	15	15	30	15	15	30	15	15	30
3	1	9,000	15	15	600	15	15	600	300	225	30
4	1	5,400	15	6	360	45	45	120	180	135	30
5	1	5,850	15	6	390	195	195	30	195	195	30
6	1	23,400	15	6	1,560	195	195	120	780	585	30
$\Sigma =$	7		105	70		1,245	1,245		2,820	2,355	

Table 4.3 Four nested block partitionings along the diagonal of Q in communications protocol model with $C = 3$

	Level 0		Level 1			Level 2			Level 3		
Partition	b_0	o_0	b_1	c_1	o_1	b_2	c_2	o_2	b_3	c_3	o_3
0	1	63,900	15	6	4,260	2,130	2,130	30	2,130	2,130	30
1	1	54,000	15	6	3,600	30	30	1,800	1,800	1,350	30
2	1	450	15	15	30	15	15	30	15	15	30
3	1	27,000	15	15	1,800	15	15	1,800	900	675	30
4	1	27,000	15	6	1,800	45	45	600	900	675	30
5	1	5,850	15	6	390	195	195	30	195	195	30
6	1	117,000	15	6	7,800	195	195	600	3,900	2,925	30
7	1	7,200	15	6	480	60	60	120	240	180	30
8	1	23,400	15	6	1,560	780	780	30	780	780	30
9	1	93,600	15	6	6,240	780	780	120	3,120	2,340	30
$\Sigma =$	10		150	78		4,245	4,245		13,980	11,265	

Kronecker structure of the communications protocol model with $C = 2, 3, 4$. Since diagonal blocks at each level become larger as C increases, we consider an alternative ordering of submodels for $C = 4$ in which submodels are ordered as 4,3,1,2. In this way, the block partitioning at level 1 yields 30 rather than 15 diagonal blocks in each reachable state space partition. As a result of this, the block partitioning at level 2 has diagonal blocks with orders not exceeding 10,000.

When Q defined over the reachable state space $\mathcal{R}$ is irreducible, its diagonal blocks all have negative diagonal entries and nonnegative off-diagonal entries. Such diagonal blocks are known to be nonsingular [30], and therefore their inverses exist. One approach that is used in block iterative methods and preconditioned projection methods [123, 243, 305] depends on factorizing diagonal blocks. In order to follow this approach, the diagonal blocks should be generated and stored as sparse matrices. The most common factorization employed in this context is LU factorization [100, 158, 302, 305] in which a given diagonal block is expressed as a product of lower-triangular and upper-

Table 4.4 Four nested block partitionings along the diagonal of Q in communications protocol model with $C = 4$ and submodels reordered as 4,3,1,2

	Level 0		Level 1			Level 2			Level 3		
Partition	b_0	o_0	b_1	c_1	o_1	b_2	c_2	o_2	b_3	c_3	o_3
0	1	144,900	30	30	4,830	30	30	4,830	450	180	322
1	1	450	30	30	15	30	30	15	450	450	1
2	1	126,000	30	10	4,200	4,200	4,200	30	63,000	25,200	2
3	1	63,000	30	10	2,100	4,200	4,200	15	63,000	63,000	1
4	1	5,850	30	30	195	30	30	195	450	180	13
5	1	81,000	30	10	2,700	1,800	1,800	45	27,000	10,800	3
6	1	351,000	30	10	11,700	1,800	1,800	195	27,000	10,800	13
7	1	23,400	30	30	780	30	30	780	450	180	52
8	1	36,000	30	10	1,200	600	600	60	9,000	3,600	4
9	1	468,000	30	10	15,600	600	600	780	9,000	3,600	52
10	1	64,800	30	30	2,160	30	30	2,160	450	180	144
11	1	9,000	30	10	300	120	120	75	1,800	720	5
12	1	259,200	30	10	8,640	120	120	2,160	1,800	720	144
$\Sigma =$	13		390	230		13,590	13,590		203,850	119,610	

triangular factors. Once the LU factors are computed, solution of the linear system involving the diagonal block boils down to forward and backward substitutions.

During LU factorization of a sparse matrix, entries that are previously zero may become nonzero. This concept is known as fill-in [305]. Ordering the rows and/or columns of a sparse matrix can result in different fill-ins in the computed [100, 158, 302, 305] LU factors [93, 130]. In order to improve the fill-in generated by LU factorization of a sparse matrix, various orderings are considered. One such effective fill-reducing ordering is *column approximate minimum degree* (`colamd`) [94, 95]. This ordering yields a permutation vector such that the LU factorization of the column permuted matrix tends to be sparser than that of the unpermuted matrix. Once column permutation of the matrix is performed according to this vector, during LU factorization row permutation may take place due to partial pivoting to ensure stability. However, the row permutation used in partial pivoting is likely to be different

than the column permutation computed by `colamd`, implying a nonsymmetric permutation.

In block partitionings of irreducible Q, the transpose of each diagonal block is column diagonally dominant. This suggests transposing each diagonal block to be LU factorized, column permuting the transposed block according to `colamd`, and using the column permutation computed by `colamd` as the row permutation in partial pivoting during its LU factorization. This has the additional advantage that all multipliers are bounded by one during LU factorization. Hence, the transposed diagonal blocks can be symmetrically permuted according to `colamd` and then LU factorized. This requires that the right-hand side is permuted before carrying out forward-backward substitutions and the resulting solution vector is inverse permuted. The `colamd` routine is available at [248].

Example 5. (cntd.) In Table 4.5, we report the number of nonzeros in LU factors of diagonal blocks of Q in the communications protocol model when original (or natural) and `colamd` orderings are used. The numbers in the table exclude nonzeros along the diagonals in all factors. The values in rows `colamd` compared to those in rows Original are quite promising.

Table 4.5 Nonzeros in LU factors of diagonal blocks of Q in communications protocol model

C	Ordering	Level 1	Level 2	Level 3
1	Original	155,070	92,370	51,870
	`colamd`	53,790	36,420	28,860
2	Original	2,201,430	785,370	397,620
	`colamd`	649,320	365,640	208,680
3	Original	25,405,245	4,763,820	1,971,180
	`colamd`	5,684,865	2,544,225	1,034,520
4	Original	85,151,670	30,769,260	3,365,100
	`colamd`	24,198,900	11,090,220	3,109,500

Grouping in MCs based on Kronecker product terms refers to collapsing the same adjacent factors in each Kronecker product. Consequently, the factors in each Kronecker product term are reduced by the same number and the state space sizes of the factors are increased. The effect of grouping factors in Kronecker product terms forming Q is investigated in a sequence of papers [26, 136, 137] under functional transitions. The objective is to reduce the number of factors and thereby the overhead associated with evaluating functional transitions. Results show that in some cases grouping may help to reduce the state space size if it had unreachable states as in each submodel of the availability model in Section 2.3, the intermediate submodels

in the production line model or each submodel in the communications protocol model of Section 3.3. Grouping may decrease the overhead associated with functional transitions and may even decrease the number of terms in the Kronecker representation. When there are functional transitions, the best approach seems to group those factors which have functional dependencies among each other. In the absence of functional transitions, it is recommended to group as many factors as possible given available memory starting from the highest indexed factor. This ensures a flatter representation for diagonal blocks at a particular level, which is a useful feature in certain iterative methods.

The effects of reordering and grouping of factors of Kronecker products on the convergence and space requirements of iterative methods have been investigated in a number of papers [58, 60, 98, 172, 315], but a broad, systematic study seems to be lacking.

4.2 Lumping

Lumpability [194] is a property possessed by some MCs which, if conditions are met, may be used to reduce a large state space to a smaller one. The idea is to find a partitioning of the original state space such that, when the states in each partition are lumped (or *aggregated*) to form a single state, the resulting MC described by the lumped states has equivalent behavior to the original chain. It is therefore important to consider lumping to reduce the size of the reachable state space, $\mathcal{R}$, before moving to the solution phase.

In this work we refer to two kinds of lumpability: ordinary lumpability and exact lumpability. Here we give definitions for CTMCs. Equivalent definitions can be stated for DTMCs. A CTMC represented by Q is said to be *ordinarily lumpable* with respect to a partitioning of its reachable state space $\mathcal{R} = \bigcup_l \mathcal{R}_l$ and $\mathcal{R}_l \bigcap \mathcal{R}_u = \varnothing$ for all $l \neq u$ if for all $\mathcal{R}_l \subset \mathcal{R}$ and all $\mathbf{i}, \mathbf{i}' \in \mathcal{R}_l$

$$\sum_{\mathbf{j} \in \mathcal{R}_u} q(\mathbf{i}, \mathbf{j}) = \sum_{\mathbf{j} \in \mathcal{R}_u} q(\mathbf{i}', \mathbf{j}) \text{ for all } \mathcal{R}_u \subset \mathcal{R} \,. \tag{4.1}$$

A CTMC represented by Q is said to be *exactly lumpable* with respect to a partitioning of its reachable state space $\mathcal{R} = \bigcup_l \mathcal{R}_l$ and $\mathcal{R}_l \bigcap \mathcal{R}_u = \varnothing$ for all $l \neq u$ if for all $\mathcal{R}_l \subset \mathcal{R}$ and all $\mathbf{i}, \mathbf{i}' \in \mathcal{R}_l$

$$\sum_{\mathbf{j} \in \mathcal{R}_u} q(\mathbf{j}, \mathbf{i}) = \sum_{\mathbf{j} \in \mathcal{R}_u} q(\mathbf{j}, \mathbf{i}') \text{ for all } \mathcal{R}_u \subset \mathcal{R} \,. \tag{4.2}$$

Ordinary lumpability refers to a partitioning of the reachable state space in which the sums of transition rates from each state in a partition to a(nother) partition are the same. On the other hand, exact lumpability refers to a partitioning of the reachable state space in which the sums of transition

rates from all states in a partition into each state of a(nother) partition are the same.

Let $\mathcal{R}_{lumped}$ denote the lumped reachable state space. On the ordinarily lumped MC, one can only compute measures defined over $\mathcal{R}_{lumped}$. On the exactly lumped MC, one can compute steady-state measures defined over $\mathcal{R}$, transient measures defined over $\mathcal{R}_{lumped}$, and transient measures defined over $\mathcal{R}$ if the states in the exactly lumpable partitions have the same initial probabilities. Since MCs satisfy a row sum property rather than a column sum property, the exact lumpability condition in (4.2) is more difficult to be satisfied than the ordinary lumpability condition in (4.1). See [44] and the references therein for more information regarding the concept of lumpability and its implications.

Lumpability can be investigated on the flat representation of the MC. Detection of ordinary and exact lumpability on Q through partition refinement would imply a time complexity of $O(nz \lg |\mathcal{R}|)$ and a space complexity of $O(nz)$ [257]. Since this is expensive in terms of time and storage, techniques that investigate lumpability on the Kronecker representation have been considered.

Lumpability can be investigated within each of the state spaces $\mathcal{S}^{(h)}$ that define the Kronecker representation of Q in (2.10) for $h = 1, \ldots, H$ independently. For the state space $\mathcal{S}^{(h)}$, detection of ordinary and exact lumpability through partition refinement as in [54] requires a time complexity of $O(nz_{Q^{(h)}} \lg n_h)$ and a space complexity of $O(nz_{Q^{(h)}})$, where

$$nz_{Q^{(h)}} = \sum_{k \in \mathcal{K}^{(h)}} nz_{Q_k^{(h)}}.$$

Then the lumped Kronecker representation may be obtained by replacing each of the state spaces $\mathcal{S}^{(h)}$ and its corresponding transition matrices $Q_k^{(h)}$ for $k = 1, \ldots, K$ with equivalent lumped ones. Lumpability can also be investigated among the state spaces $\mathcal{S}^{(h)}$ that are *replicated* (or identical) with respect to the Kronecker representation of Q as in [24]. Therein ordinary lumpability of replicated state spaces is shown in the presence of functional transitions over the product state space $\mathcal{S}$ in SANs. Note that replication refers to a very specific kind of symmetry in the Kronecker representation, and with ordinary lumpability, only measures of interest over $\mathcal{S}_{lumped}$ can be computed. Lumpability can also be investigated among the state spaces $\mathcal{S}^{(h)}$ by considering dependencies and matrix properties in the Kronecker representation as in [174, 175]. Therein sufficient conditions that satisfy ordinary lumpability over $\mathcal{S}$ in SANs are specified and an iterative steady-state solution method which is able to compute measures over $\mathcal{S}$ is given for CTMCs and DTMCs in the presence of functional transitions. The work identifies lumpable partitionings on the underlying MC induced by the nested block structure of the Kronecker representation in (2.3). Although the particular approach of lumping one or more state spaces $\mathcal{S}^{(h)}$ totally as in [174, 175]

is a very specific kind of performance equivalence and lumping considered in [45, 51], due to its accommodation of functional transitions, it also enables the detection of certain ordinarily lumpable partitionings in which blocks are composed of multiple (nonidentical) state spaces, but the individual state spaces cannot be lumped by themselves. This is not possible with the approaches in [24, 45, 51].

We remark that neither of the two approaches in [24] and [174, 175] that investigate lumpability among the state spaces $\mathcal{S}^{(h)}$ for $h = 1, \ldots, H$ is completely automated, uses a Kronecker representation for the lumped MC, and possesses a proper complexity analysis. Furthermore, since the Kronecker representation is rich in structure and the three approaches presented in this chapter do not work on the flat representation, there can very well be other symmetries in the Kronecker representation which also lead to lumpability. Along a slightly different line, the concept of *quasi-lumpability* [122] can be investigated.

4.3 Analyzing Diagonal Blocks for Common Schur Factors

One of the things Chapter 2 has shown is that the Kronecker structure of Q in (2.10) underlying an H-dimensional CTMC induces nested block partitionings. In each diagonal block $Q(p,p)$ for $p = 0, \ldots, N-1$, there may be diagonal blocks at levels 1 through $H-1$ with identical off-diagonal parts and diagonals differing from each other by a multiple of the identity matrix. Such diagonal blocks are named candidate blocks [58] in the following, which explains how they can be detected and mutually benefit from a single real Schur factorization [125, 303]. The objective is to reduce the storage required by the factors of the diagonal blocks and the solution time in iterative solvers. We specify sufficient conditions for the existence of diagonal blocks with real eigenvalues that can be checked using the interaction matrix and submodel transition matrices $Q_k^{(h)}$. This is a significant result, because in practice the real Schur factors of diagonal blocks satisfying these conditions are sparse. Furthermore, as we will describe, these factors can be constructed from $Q_k^{(h)}$ and their real Schur factors.

When Q is in Kronecker form, its diagonal blocks possess various properties due to the nested block partitioning. Let us first look at the difference between diagonal blocks of Kronecker product terms due to local transitions and synchronizing transitions. Diagonal blocks at each level of Kronecker products with all factors except one being I are all identical up to their diagonals, and the diagonals differ by a multiple of I. In other words, diagonal blocks of Kronecker products associated with local transitions automatically satisfy the sought property. Therefore, in each diagonal block $Q(p,p)$ for $p = 0, \ldots, N-1$, in order to detect diagonal blocks with identical off-

diagonal parts and diagonals differing from each other by a multiple of I, it suffices to check conditions related to Kronecker products with at least two nonidentity factors (i.e., Kronecker products corresponding to synchronizing transitions). Hereafter, diagonal blocks satisfying the described property are named *candidate blocks*.

The set of diagonal blocks in $Q(p,p)$ may form multiple partitions of candidate blocks, where blocks in each partition test positively for candidacy, but two blocks in different partitions fail the test either due to having different off-diagonal parts or their diagonals not differing from each other by a multiple of I. Since Kronecker products contribute to diagonal blocks in various ways, detecting all such partitions is a difficult task. The next algorithm, Algorithm 3, presented in [58] may not detect all candidate blocks, but we will be content with those that get detected since it executes rapidly and we do not want to compute multiple real Schur factorizations within $Q(p,p)$. The algorithm can be used to check candidacy among all diagonal blocks in $Q(p,p)$ for $p = 0,\ldots,N-1$ at block partitioning level $l = 1,\ldots,H-1$. The two conditions C1 and C2 together are sufficient to detect candidate blocks, and they can be checked using the interaction matrix and $Q_k^{(h)}$ which are held in row sparse format.

ALGORITHM 3. *Detecting candidate blocks in $Q(p,p)$ at level l.*

For each Kronecker product that contributes to $Q(p,w)$ for $w := 0,\ldots,N-1$:

C1. In $Q(p,p)$ at level l, make sure there is:
(a) either no contribution from the Kronecker product to the candidate block, or
(b) the same contribution from the Kronecker product to all diagonal blocks (including the candidate block), or
(c) a contribution from the Kronecker product that is a multiple of I to some diagonal blocks (including the candidate block);

C2. In $Q(p,w)$ for $p \neq w$ at level l, make sure there is:
(a) either no contribution from the Kronecker product to the off-diagonal blocks in the same row as the candidate block, or
(b) the same contribution from the Kronecker product to the diagonals of all diagonal blocks (including the candidate block), or
(c) a contribution from the Kronecker product that is a multiple of I to the diagonals of some diagonal blocks (including the candidate block).

Example 5. (ctnd.) For instance, in reachable state space partition $\mathcal{R}_0$ of the communications protocol model with $C = 1$, Kronecker products due to transitions 5 in $Q_{0,0}$, 7 in $Q_{0,1}$, 6 in $Q_{0,2}$, and 6 in $Q_{0,3}$ must be checked (see the interaction matrix in (3.2)).

C1 in Algorithm 3 ensures that off-diagonal parts of all candidate blocks are identical and their diagonals differ from each other by a multiple of I.

Specifically, C1(b) and C1(c) are checked by inspecting $Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})$, respectively, up to and after block partitioning level l in the Kronecker product and making sure they are all equal to I. C2 ensures after C1 is satisfied that the contribution from off-diagonal blocks to diagonals of candidate blocks do not change the fact that diagonals of candidate blocks differ from each other by a multiple of I. Specifically, C2(b) and C2(c) are checked by inspecting $Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$, respectively, up to and after block partitioning level l and making sure they have equal row sums.

Example 5. (cntd.) Tables 4.1 through 4.4 list the number of candidate blocks under column c_l for $l = 1, 2, 3$ in the communications protocol model for different block partitioning levels obtained by executing Algorithm 3. Note that the block partitioning at level 0 is omitted since it yields one diagonal block in each reachable state space partition. At the other block partitioning levels, there is a relatively large number of candidate blocks; the partitioning at level 2 even yields all diagonal blocks as candidates.

Now, we explain how candidate blocks in $Q(p, p)$ can all take advantage of the real Schur factorization of one of the candidate blocks. The *real Schur factorization* is an eigenvalue revealing factorization and exists for real non-symmetric square matrices [158, 303]. If $A \in \mathbb{R}^{o_l \times o_l}$, then it can be written as $A = VTV^T$, where V is orthogonal (i.e., $V^TV = I$) and T is *quasi-triangular*, meaning it is block triangular with blocks of order 1 or 2 along its diagonal. The columns of V contain the real Schur vectors. The blocks of order 1 along the diagonal of T contain the real eigenvalues of A, and the blocks of order 2 along the diagonal of T contain the pairs of complex conjugate eigenvalues of A. When both T and V are to be obtained, the cost of factorizing A of order o_l into real Schur form, assuming it is full, is $25o_l^3$ [125]. Note that A can also be in the form $A = (VE)(E^TTE)(VE)^T$ for a permutation matrix E in which E^TTE is quasi-triangular. Without loss of generality, we assume that T is in quasi-upper-triangular form.

Real Schur factorization of a matrix can be obtained using the LAPACK routine `dgees` [125] available at [248]. This routine uses two two-dimensional floating-point arrays the first of which has matrix A to be real Schur factorized on input and (quasi-)upper-triangular factor T on output, whereas the second has orthogonal factor V on output. The returned real Schur factors T and V can be compacted and stored as sparse matrices to be used in iterative solvers. The storage that is set aside for the two two-dimensional floating-point arrays in the preprocessing phase is temporary and not used in iterative solvers.

Now, let A_1 be the first candidate block in $Q(p, p)$, and A_i, $i > 1$, denote the ith candidate block in $Q(p, p)$. Let $A_i - A_1 = \lambda_i I$ and the real Schur form of A_1 be given by $A_1 = VTV^T$. Then $A_i = V(T + \lambda_i I)V^T$. Hence, if we are to solve $p_iA_i = b_i$, where p_i and b_i are row subvectors of appropriate length, then we can do so by solving the equivalent system $p_iV(T + \lambda_i I)V^T = b_i$ in three steps: 1) compute $c_i := b_iV$; 2) let $y_i = p_iV$; solve $y_i(T + \lambda_i I) = c_i$

for y_i; 3) compute $p_i := y_i V^T$. These three steps require two vector–matrix multiplications and one quasi-triangular solve. All that needs to be done is to store λ_i for each candidate block and the real Schur factors T and V so that they are accessible from $Q(p,p)$ later in the solution process. To expedite this process, we choose to store the reciprocals of the diagonals of the matrices $(T + \lambda_i I)$. This exchanges the addition and division per diagonal entry at each real Schur solve with a one time division and a per solve multiplication per diagonal entry.

Example 5. (cntd.) Table 4.6 reports the total number of nonzeros in diagonal blocks when all candidate blocks in $Q(p,p)$ at the designated level for $p = 0,\ldots,N-1$ use common real Schur factors, and the remaining diagonal blocks use LU factors. The numbers in the table exclude the λ_i's and the nonzeros along the diagonals in all factors. For diagonal blocks that use LU factors, two orderings, Original and `colamd`, are considered as in Table 4.5. The results indicate substantial savings over those in Table 4.5 since the real

Table 4.6 Nonzeros in factors of candidates and remaining diagonal blocks of Q in communications protocol model

C	Ordering	Level 1		Level 2		Level 3	
		LU	Schur	LU	Schur	LU	Schur
1	Original	78,957	4,052	0	1,498	5,985	532
	`colamd`	27,432	4,052	0	1,498	3,330	532
2	Original	1,245,411	32,154	0	8,403	65,565	931
	`colamd`	352,278	32,154	0	8,403	34,410	931
3	Original	14,968,098	173,005	0	30,436	382,815	1,330
	`colamd`	3,220,922	173,005	0	30,436	200,910	1,330
4	Original	50,312,460	344,569	0	66,178	2,243,430	3,366
	`colamd`	14,297,760	344,569	0	66,178	2,077,110	3,366

Schur factors turn out to be quite sparse compared to the LU factors. We noticed that this is a property of candidate blocks having real eigenvalues (i.e., triangular T factor). This hints at a reduction in storage and possibly in solution time. On the other hand, we noticed that when the candidate block has complex eigenvalues, the real Schur factors are either nearly or completely full. For the real Schur factorization to be worthwhile the effort, it should not yield excessive number of nonzeros in the factors.

Let us now state sufficient conditions for the existence of diagonal blocks with real eigenvalues in $Q(p,p)$. The results are based on the fact that the Kronecker product and the Kronecker sum of two nonsymmetric square matrices with real eigenvalues have real eigenvalues and therefore are triangularizable using orthogonal matrices. See [58] for a proof of this result.

First, we let the real Schur factorization of local transition submatrix of submodel h associated with $\mathcal{K}_{p,p}$ be given by

$$Q_h^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) = V_h T_h {V_h}^T,$$

where T_h is (quasi-)upper-triangular and V_h is orthogonal. Observe that $Q(p,p)$ is a sum of three terms: the first due to the Kronecker sum of submodel local transition submatrices associated with $\mathcal{K}_{p,p}$, the second due to the sum of Kronecker products of submodel synchronizing transition submatrices associated with $\mathcal{K}_{p,p}$, and the third due to the diagonal correction necessary to have $Q(p,p)\mathbf{e} = -\sum_{w=0}^{p-1} Q(p,w)\mathbf{e} - \sum_{w=p+1}^{N-1} Q(p,w)\mathbf{e}$. Let $\tilde{Q}_D(p,p)$ denote the diagonal block of $Q_D(p,p)$ associated with the particular diagonal block under consideration for real Schur factorization.

Let us state sufficient condition 1 (SC1) for the existence of real eigenvalues in diagonal blocks of $Q(p,p)$ from [58]. If

(a) $\mathcal{K}_{p,p}$ does not possess synchronizing transitions, and
(b) each T_h in $Q_h^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) = V_h T_h {V_h}^T$ for $h > l$ is upper-triangular, and
(c) $(\bigotimes_{h>l} {V_h}^T)\tilde{Q}_D(p,p)(\bigotimes_{h>l} V_h)$ is diagonal,

then the diagonal block under consideration at level l in $Q(p,p)$ has real eigenvalues.

We remark that SC1(a) is satisfied by all $Q(p,p)$ for $p = 0, \ldots, N-1$ in many models arising from closed queueing networks. SC1(b) is satisfied, for instance, when $Q_h^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})$ for submodels $h > l$ are triangular. SC1 also describes an approach to construct the factors of the diagonal block that is to be real Schur factorized at level l in $Q(p,p)$ from the real Schur factors of $Q_h^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})$ and $\tilde{Q}_D(p,p)$. Indeed,

$$V = \bigotimes_{h>l} V_h \quad \text{and} \quad T = \bigoplus_{h>l} T_h + (\bigotimes_{h>l} {V_h}^T)\tilde{Q}_D(p,p)(\bigotimes_{h>l} V_h).$$

It is also possible to check SC1(c) and build the product therein using the orthogonal real Schur factors of $Q_h^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})$ as explained in [58]. Here, we provide the result.

If SC1 holds at level l in $Q(p,p)$ and all V_h for $h > l$ are permutation matrices, then the orthogonal real Schur factor of the diagonal block under consideration for real Schur factorization at that level is a permutation matrix, and its upper-triangular factor has the same number of nonzeros as the diagonal block. In other words, real Schur factorization will amount to computing a permutation matrix that upper-triangularizes the particular diagonal block. A similar result holds for the next condition SC2, which subsumes SC1 but is given separately, because, as we will see in the communications protocol model, there may be reachable state space partitions satisfying SC1 and others satisfying SC2.

Let us state sufficient condition 2 (SC2) for the existence of real eigenvalues in diagonal blocks in $Q(p,p)$ from [58]. If

(a) SC1(b), and

(b) each $\bigotimes_{h>l}({V_h}^T Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})V_h)$ that contributes to the diagonal block under consideration at level l for all synchronizing transitions $k \in \mathcal{K}_{p,p}$ is upper-triangular, and,

(c) SC1(c),

then the diagonal block under consideration at level l in $Q(p,p)$ has real eigenvalues.

The proof of SC2 is similar to that of SC1, and it is possible to construct the real Schur factors of the particular diagonal block at level l in $Q(p,p)$ from the real Schur factors of $Q_h^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)})$ for local transitions $h > l$, $Q_k^{(h)}(\mathcal{S}_p^{(h)}, \mathcal{S}_p^{(h)})$ for all synchronizing transitions $k \in \mathcal{K}_{p,p}$ and submodels $h > l$, and $\tilde{Q}_D(p,p)$. Checking SC2(b) requires one to have previously computed the multipliers that multiply each Kronecker product in forming the candidate block when $l > 0$. However, this is something we do in detecting candidate blocks and use in step C1(a) of Algorithm 3. In other words, those Kronecker products that will eventually get multiplied with a zero and therefore do not contribute to the diagonal block under consideration should not change our decision regarding the satisfiability of SC2. Note also that it suffices for the first nondiagonal factor in the Kronecker product of SC2(b) to be an upper-triangular matrix to satisfy the condition for the particular synchronizing transition $k \in \mathcal{K}_{p,p}$ (see Appendix A in [315]).

When $Q(p,p)$ satisfies SC1 or SC2, real Schur factors of the first candidate block are constructed as described from specific submodel submatrices. This is the Kronecker-based sparse approach. When SC1 and SC2 are not satisfied, it is still possible to compute real Schur factors of the particular candidate block using `dgees`. This is the flat approach. We remark that the real Schur factors computed using the Kronecker-based sparse approach and the flat approach can be different but will be related to each other by a permutation matrix, E, as mentioned before. Nevertheless, their products always give the same diagonal block. The Kronecker-based sparse approach is expected to require less temporary storage and be faster than the flat approach as the order of the candidate block to be real Schur factorized increases.

Example 5. (cntd.) In the communications protocol model with $C = 1$, all $Q(p,p)$ for $p = 0, \ldots, 3$ except $Q(1,1)$ satisfy SC2 and $Q(1,1)$ satisfies SC1 (see the interaction matrix in (3.2) and consider the submodel submatrices that contribute to $Q(p,p)$) for nested block partitioning levels $l = 1, 2, 3$. In each $Q(p,p)$ for $p = 0, \ldots, 3$, the V factor is a permutation matrix, and the T factor has as many nonzeros as in the candidate blocks.

We emphasize that even though l may be small and diagonal blocks at that level large, both SC1 and SC2 require the real Schur factorization of submodel local transition submatrices only. Even then, real Schur factorization needs to be employed only when the input matrix is not triangular. And when either SC1 or SC2 is satisfied, we have a very efficient way of constructing the real Schur factors of candidate blocks.

4.4 Handling Countable Infiniteness for Steady-State

In this section, we discuss how the countably infinite reachable state space of a MC can be truncated for steady-state analysis [157, 314]. Truncation approaches for transient analysis are considered in Chapter 7. We motivate the discussion with another specification technique for multidimensional Markovian models, which represents classes of transitions among states by using state change vectors. While we give a formal definition of transition classes [184] that can be used in Kronecker-based modeling of systems of stochastic chemical kinetics in the next subsection, for the time being, we will be content with the informal definition provided next.

Given a state vector $\mathbf{i} \in \mathbb{Z}_{\geqslant 0}^{1\times H}$ representing the current state of an H-dimensional Markovian model, a *state change vector* $\mathbf{v}^{(k)} \in \mathbb{Z}^{1\times H}$ and a transition rate $\alpha_k : \mathcal{R} \mapsto \mathbb{R}_{\geqslant 0}$, the model makes a transition of class k to state $(\mathbf{i} + \mathbf{v}^{(k)})$ with rate $\alpha_k(\mathbf{i})$ when $\mathbf{i} \in \mathcal{R}$, $\alpha_k(\mathbf{i}) > 0$, and $\mathbf{i} + \mathbf{v}^{(k)} \geqslant \mathbf{0}$ for $k = 1, \ldots, K$. Here, $\mathcal{R}$ is the reachable state space of the model, $\mathbf{v}^{(k)} \neq \mathbf{0}$, and $\alpha_k(\mathbf{i})$ is a multivariate function in the state variables i_h for $h = 1, \ldots, H$ and $k = 1, \ldots, K$. It should be apparent from the values $\mathbf{i} \in \mathcal{R}$ can take that reachable state space $\mathcal{R}$ is countably infinite and needs to be truncated judiciously for analysis purposes. The following approach enables us to identify those states on which a certain amount of the steady-state probability mass is concentrated when $\mathcal{R}$ is irreducible.

An irreducible CTMC with countably infinite reachable state space $\mathcal{R}$ is ergodic and has a steady-state distribution if and only if there exists a *Lyapunov function* $g : \mathcal{R} \to \mathbb{R}_{\geqslant 0}$ and a finite set $\mathcal{C} \subset \mathcal{R}$ simultaneously satisfying the three conditions: [314]

(i) $d(\mathbf{i}) \leqslant -\gamma$ for all $\mathbf{i} \in \mathcal{R}\backslash\mathcal{C}$ and some $\gamma > 0$,

(ii) $d(\mathbf{i}) < \infty$ for all $\mathbf{i} \in \mathcal{C}$,

(iii) $\{\mathbf{i} \in \mathcal{R} \mid g(\mathbf{i}) \leqslant r\}$ is finite for all $r < \infty$,

where

$$d(\mathbf{i}) = \sum_{k=1}^{K} \alpha_k(\mathbf{i})(g(\mathbf{i} + \mathbf{v}^{(k)}) - g(\mathbf{i})) \in \mathbb{R} \tag{4.3}$$

is the *drift* in state $\mathbf{i} \in \mathcal{R}$. In other words, the drift must be negative for the countably infinite number of states in reachable state space $\mathcal{R}$ excluding those in $\mathcal{C}$, the drift must be finite for the states in $\mathcal{C}$, and the set of states for which the Lyapunov function attains a finite value must always be finite. Note that the Lyapunov function is assumed to be nonnegative in [314], but the above result is also valid if the Lyapunov function is bounded from below as indicated in [131].

When there is a candidate Lyapunov function $g(\mathbf{i})$ satisfying condition (iii) at hand and

$$\chi = \sup_{\mathbf{i}\in\mathcal{R}} d(\mathbf{i}) < \infty$$

(note that the latter condition is equivalent to satisfying conditions (i) and (ii) simultaneously), it is possible to specify

$$\varepsilon = \chi/(\chi + \gamma) \in (0, 1)$$

as an upper bound on $\sum_{\mathbf{i} \in \mathcal{R} \setminus \mathcal{C}} \boldsymbol{\pi}(\mathbf{i})$. Equivalently, it is possible to use

$$\gamma = \chi/\varepsilon - \chi$$

in constructively defining

$$\mathcal{C} = \{\mathbf{i} \in \mathcal{R} \mid -\gamma < d(\mathbf{i}) < \infty\}. .$$

Now, if it is further shown that $\mathcal{C}$ is finite, then the three conditions above hold and

$$\sum_{\mathbf{i} \in \mathcal{C}} \boldsymbol{\pi}(\mathbf{i}) \geqslant 1 - \varepsilon.$$

However, in many cases it is not a trivial task to show the finiteness of $\mathcal{C}$ for a chosen Lyapunov function $g(\mathbf{i})$. This seems to be an open problem and worth exploring.

Clearly, determining χ is also a problem in itself. To that end, the domain of the search for extrema should be restricted to $\mathbb{R}_{\geqslant 0}^{1 \times H}$. All extrema can then be computed by equating the gradient of $d(\mathbf{i})$ to zero. In order to determine all local extrema including those located on the boundaries of the domain, the same system must be solved for every projection of $d(\mathbf{i})$ onto each subspace of $\mathbb{R}^{1 \times H}$ by setting all combinations of state variables i_h for $h = 1, \ldots, H$ to 0. In the end, all extrema outside $\mathbb{R}_{\geqslant 0}^{1 \times H}$ should be discarded. Throughout this process, the resulting systems of nonlinear equations can be solved, for instance, using the HOM4PS-2.0 package [215], which implements the polyhedral homotopy continuation method.

Once the finiteness of $\mathcal{C}$ is proved and χ (or equivalently, γ for chosen ε) is determined, it is possible to form the set $\mathcal{C}$. Clearly, the quality of the approximations increases as the steady-state probability mass concentrated on states in $\mathcal{C}$ approaches 1 (i.e., as ε approaches 0) [120].

4.4.1 A Metabolite Synthesis Model

A *system of stochastic chemical kinetics* [31, 208, 238, 254, 296, 300, 318] describes the dynamics of a number of molecules that interact through chemical reactions. The state space of the corresponding Markovian model considered here is discrete, countably infinite, and H-dimensional [120]. Among the H state variables, H_I are countably infinite and represent molecule numbers, while H_F are finite and represent the finite control mechanism by which molecules interact. Since the finite control mechanism may or may

not exist, it follows that $H_I \geqslant 2$, $H_F \geqslant 0$, and $H = H_I + H_F$. Without loss of generality, we assume that the first H_I state variable indices are assigned to the molecules. Hence, the state spaces of the variables satisfy $|\mathcal{S}^{(h)}| \to \infty$ for $h = 1, \ldots, H_I$ and $|\mathcal{S}^{(h)}| < \infty$ for $h = H_I + 1, \ldots, H$. Now, let $\bar{\mathcal{R}} \subseteq \times_{h=H_I+1}^{H} \mathcal{S}^{(h)}$ denote the set of states the finite variables can take.

Where it is the number of molecules that is relevant in chemical kinetics, in other domains it may be the number of species, genes, proteins, etc. Therefore, we will be using these terms interchangeably. The discrete Markovian model considered here is more realistic and shown to behave differently than the respective deterministic model, which is continuous and in the form of a system of ordinary differential equations [156, 208]. This is especially true when the numbers of different types of molecules are relatively small. We recall that in many cases not all states in the product state space $\mathcal{S} = \times_{h=1}^{H} \mathcal{S}^{(h)}$ are necessarily reachable. However, each state in the reachable state space $\mathcal{R}$ that satisfies $\mathcal{R} \subseteq \mathcal{S}$ is reachable from every other state since the associated CTMC is assumed to be irreducible for steady-state analysis.

Transition class k is a pair

$$(\alpha_k(\mathbf{i}), \mathbf{v}^{(k)}) \,,$$

where $\alpha_k(\mathbf{i}) \in \mathbb{R}_{\geqslant 0}$ is the *transition rate function* and $\mathbf{v}^{(k)} \in \mathbb{Z}^{1\times H}$ is the *state change vector* for $k = 1 \ldots, K$ [103, 120]. The first element of the pair specifies the transition rate from state $\mathbf{i} \in \mathcal{R}$ to state $(\mathbf{i}+\mathbf{v}^{(k)}) \in \mathcal{R}$. The second element of the pair specifies the successor state of the transition, where $v_h^{(k)}$ denotes the change in state variable i_h due to a class k transition.

When the transition rate function $\alpha_k(\mathbf{i})$ for transition class k is *separable*, it may be written as

$$\alpha_k(\mathbf{i}) := \phi_k \prod_{h=1}^{H} \alpha_k^{(h)}(i_h) \,, \tag{4.4}$$

where $\phi^{(k)} \in \mathbb{R}_{>0}$ is its *state independent transition rate* and $\alpha_k^{(h)}(i_h) : \mathcal{S}^{(h)} \mapsto \mathbb{R}_{\geqslant 0}$ is its *state dependent transition rate* for variable i_h with $h = 1, \ldots, H$ [107]. The separability of the transition rate function $\alpha_k(\mathbf{i})$ into the scalar ϕ_k times H factors $\alpha_k^{(h)}(i_h)$ for $h = 1, \ldots, H$ holds in elementary reactions which are building blocks for more complicated reaction kinetics [126, 156, 188, 236, 283]. Typically the number of reactions in a system of stochastic chemical kinetics is finite, and each reaction corresponds to one transition class.

Consider the model of a biological process of metabolite synthesis [218] with repressilator in Figure 4.1 having the transition classes in Table 4.7. Here, $H = 6$, $H_I = 3$, $H_F = 3$, $\mathbf{i} = (i_1, i_2, i_3, i_4, i_5, i_6)$, $K = 12$, and λ_1, λ_2, λ_3, μ_1, μ_2, μ_3, β_0, $\beta_1 \in \mathbb{R}_{>0}$. The model has three types of genes G_1, G_2, G_3 and three different control variables r_1, r_2, r_3. The genes regulate each other's synthesis in a cyclic manner, G_1 regulating G_2, G_2 regulating G_3, and

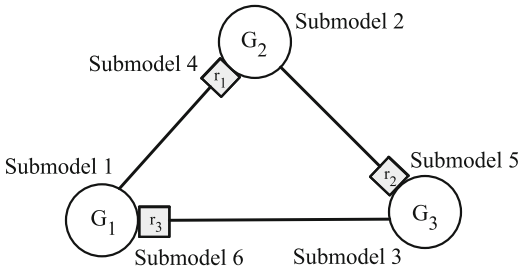


Fig. 4.1 Metabolite synthesis model with six submodels

G_3 regulating G_1. Each type of gene has a single binding site to which only one repressor at a time can bind. A gene regulates another type of gene by producing its own type of repressor, the repressor binding to the binding site of the gene to be regulated, and thereby repressing (or blocking) the other

Table 4.7 Transition classes of the metabolite synthesis model with repressilator

k	ϕ_k	$\alpha_k^{(1)}(i_1)$	$\alpha_k^{(2)}(i_2)$	$\alpha_k^{(3)}(i_3)$	$\alpha_k^{(4)}(i_4)$	$\alpha_k^{(5)}(i_5)$	$\alpha_k^{(6)}(i_6)$	$\mathbf{v}^{(k)}$
1	λ_1	1	1	1	1	1	$(1-i_6)$	$\mathbf{e}_1^T$
2	μ_1	i_1	1	1	1	1	1	$-\mathbf{e}_1^T$
3	β_0	i_1	1	1	$(1-i_4)$	1	1	$(-\mathbf{e}_1+\mathbf{e}_4)^T$
4	β_1	1	1	1	i_4	1	1	$(\mathbf{e}_1-\mathbf{e}_4)^T$
5	λ_2	1	1	1	$(1-i_4)$	1	1	$\mathbf{e}_2^T$
6	μ_2	1	i_2	1	1	1	1	$-\mathbf{e}_2^T$
7	β_0	1	i_2	1	1	$(1-i_5)$	1	$(-\mathbf{e}_2+\mathbf{e}_5)^T$
8	β_1	1	1	1	1	i_5	1	$(\mathbf{e}_2-\mathbf{e}_5)^T$
9	λ_3	1	1	1	1	$(1-i_5)$	1	$\mathbf{e}_3^T$
10	μ_3	1	1	i_3	1	1	1	$-\mathbf{e}_3^T$
11	β_0	1	1	i_3	1	1	$(1-i_6)$	$(-\mathbf{e}_3+\mathbf{e}_6)^T$
12	β_1	1	1	1	1	1	i_6	$(\mathbf{e}_3-\mathbf{e}_6)^T$

type of gene. This kind of binding in which only one repressor can bind to a binding site and repress a type of gene is said to be noncooperative. When the first control variable is set to 1 (i.e., $i_4 = 1$), a type 1 repressor r_1 is bound to the binding site of G_2. Similarly, when $i_5 = 1$, a type 2 repressor r_2 is bound to the binding site of G_3, and when $i_6 = 1$, a type 3 repressor r_3 is bound to the binding site of G_1. When a control variable is set to 0 rather than 1, the corresponding binding site is unbound. Hence, we have

$$\mathcal{S}^{(1)} = \mathcal{S}^{(2)} = \mathcal{S}^{(3)} = \mathbb{Z}_{\geqslant 0}, \quad \mathcal{S}^{(4)} = \mathcal{S}^{(5)} = \mathcal{S}^{(6)} = \{0,1\},$$

$$\bar{\mathcal{R}} = \mathcal{S}^{(4)} \times \mathcal{S}^{(5)} \times \mathcal{S}^{(6)},$$

$|\bar{\mathcal{R}}| = 8$, and $\mathcal{R} = \mathcal{S}^{(1)} \times \mathcal{S}^{(2)} \times \mathcal{S}^{(3)} \times \bar{\mathcal{R}}$. Parameters λ_h and μ_h for $h = 1, 2, 3$, respectively, denote state-independent production and degradation rates of type h genes, whereas parameters β_0 and β_1, respectively, denote state-independent binding and unbinding rates. However, degradation and binding rates are in fact state dependent. In this model, reachable state space $\mathcal{R}$ is equal to the product state space.

Having specified the parameters of the model when the separability condition in (4.4) holds, two more things need to be done to formulate its Kronecker representation. First, for each transition class, the transition matrices associated with state variables must be specified, one matrix per state variable. Second, these transition matrices must be composed using the Kronecker product operator and the state-independent transition rates to form the blocks. It is these problems that we now consider.

We first associate transition matrices with countably infinite state variables. Observe that there are H_I such transition matrices to be associated with each transition class. The transition matrix of state variable i_h for $h = 1, \ldots, H_I$ and transition class $k = 1, \ldots, K$ is denoted by $Q_k^{(h)} \in \mathbb{R}_{\geqslant 0}^{|\mathcal{S}^{(h)}| \times |\mathcal{S}^{(h)}|}$ and given entrywise as

$$Q_k^{(h)}(i_h, j_h) = \begin{cases} \alpha_k^{(h)}(i_h) & \text{if } j_h = i_h + v_h^{(k)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i_h, j_h \in \mathcal{S}^{(h)} .$$

Regarding finite state variables when $H > H_I$, we prefer to define a grouped transition matrix since we have observed in practice that $|\mathcal{S}^{(h)}|$ for $h = H_I + 1, \ldots, H$ is very small. Recalling that $\bar{\mathcal{R}}$ denotes the set of states finite variables can take and therefore $\bar{\mathcal{R}} \subseteq \times_{h=H_I+1}^{H} \mathcal{S}^{(h)}$, the grouped transition matrix of finite state variables for transition class $k = 1, \ldots, K$ is denoted by $\bar{Q}_k \in \mathbb{R}_{\geqslant 0}^{|\bar{\mathcal{R}}| \times |\bar{\mathcal{R}}|}$ and is given entrywise as

$$\bar{Q}_k((i_{H_I+1}, \ldots, i_H), (j_{H_I+1}, \ldots, j_H))$$

$$= \begin{cases} \prod_{h=H_I+1}^{H} \alpha_k^{(h)}(i_h) & \text{if } (j_{H_I+1}, \ldots, j_H) = (i_{H_I+1}, \ldots, i_H) \\ & \qquad + (v_{H_I+1}^{(k)}, \ldots, v_H^{(k)}) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for } (i_{H_I+1}, \ldots, i_H), (j_{H_I+1}, \ldots, j_H) \in \bar{\mathcal{R}}.$$

When $H = H_I$, it is assumed that $|\bar{\mathcal{R}}| = 1$ and $\bar{Q}_k = (1)$.

The transition matrices corresponding to the countably infinite state variables of the model in Table 4.7 are obtained as

$$
\begin{aligned}
Q_1^{(2)} &= Q_1^{(3)} = Q_2^{(2)} = Q_2^{(3)} = Q_3^{(2)} = Q_3^{(3)} = Q_4^{(2)} = Q_4^{(3)} = Q_5^{(1)} = Q_5^{(3)} \\
&= Q_6^{(1)} = Q_6^{(3)} = Q_7^{(1)} = Q_7^{(3)} = Q_8^{(1)} = Q_8^{(3)} = Q_9^{(1)} = Q_9^{(2)} \\
&= Q_{10}^{(1)} = Q_{10}^{(2)} = Q_{11}^{(1)} = Q_{11}^{(2)} = Q_{12}^{(1)} = Q_{12}^{(2)} = I_\infty \ , \\
Q_1^{(1)} &= Q_4^{(1)} = Q_5^{(2)} = Q_8^{(2)} = Q_9^{(3)} = Q_{12}^{(3)} = \mathrm{supdiag}((1,1,\ldots)^T) \ , \\
Q_2^{(1)} &= Q_3^{(1)} = Q_6^{(2)} = Q_7^{(2)} = Q_{10}^{(3)} = Q_{11}^{(3)} = \mathrm{subdiag}((1,2,\ldots)^T) \ ,
\end{aligned}
$$

The grouped transition matrices corresponding to finite state variables of the same model are obtained as

$$
\begin{aligned}
\bar{Q}_1 &= I_2 \otimes I_2 \otimes \mathrm{diag}((1,0)^T) \ , \quad \bar{Q}_2 = \bar{Q}_6 = \bar{Q}_{10} = I_2 \otimes I_2 \otimes I_2 \ , \\
\bar{Q}_3 &= \mathrm{supdiag}((1)^T) \otimes I_2 \otimes I_2 \ , \quad \bar{Q}_4 = \mathrm{subdiag}((1)^T) \otimes I_2 \otimes I_2 \ , \\
\bar{Q}_5 &= \mathrm{diag}((1,0)^T) \otimes I_2 \otimes I_2 \ , \quad \bar{Q}_7 = I_2 \otimes \mathrm{supdiag}((1)^T) \otimes I_2 \ , \\
\bar{Q}_8 &= I_2 \otimes \mathrm{subdiag}((1)^T) \otimes I_2 \ , \quad \bar{Q}_9 = I_2 \otimes \mathrm{diag}((1,0)^T) \otimes I_2 \ , \\
\bar{Q}_{11} &= I_2 \otimes I_2 \otimes \mathrm{supdiag}((1)^T) \ , \quad \bar{Q}_{12} = I_2 \otimes I_2 \otimes \mathrm{subdiag}((1)^T) \ .
\end{aligned}
$$

Our objective is to formulate a Kronecker representation for the nonzero blocks of Q from the transition matrices and the state-independent transition rates. To this end, let us start by formally defining $\mathcal{R}_p$ to be the subset of states corresponding to partition $p \in \mathbb{Z}_{\geqslant 0}$ given by

$$
\begin{aligned}
\mathcal{R}_p &= \{\mathbf{i} \in \mathcal{R} \mid \max_{h=1,\ldots,H_I}(i_h) = p\} \ , \\
\mathcal{R} &= \bigcup_{p=0}^{\infty} \mathcal{R}_p, \ \mathcal{R}_p \cap \mathcal{R}_w = \varnothing \quad \text{for } p \neq w \ .
\end{aligned} \tag{4.5}
$$

The maximum function is justified by observing that the maximum valued variable among $i_1, \ldots, i_{H_I}$ in any state $\mathbf{i} \in \mathcal{R}$ changes by at most one through any transition due to the particular form of the state change vectors $\mathbf{v}^{(k)}$ in the transition classes for systems of stochastic chemical kinetics. As a by-product of this, Q turns out to be block tridiagonal.

Recall that the set operation associated with Kronecker product is Cartesian product. As we will see, each reachable state space partition needs to be expressed as a union of Cartesian products of state space partitions of the countably infinite variables for the class of models considered. Therefore, reachable state space partition index definitions, such as $p = \sum_{h=1}^{H_I} i_h$, that have arithmetic dependencies among countably infinite variables seem to be less suitable in trying to come up with a Kronecker representation.

For each reachable state space partition $\mathcal{R}_p$, the values a variable can take depend on the values of other variables. Therefore, first we define a partition of the values a countably infinite variable can take where there is no such dependency in a way similar to HMMs [43]. Then we introduce subpartitions of $\mathcal{R}_p$ based on the partitions of countably infinite variables defined before. Let

$$\mathcal{S}_{p,l}^{(h)} = \begin{cases} \{i_h \mid 0 \leqslant i_h \leqslant p\} & \text{if } l < h \\ \{p\} & \text{if } l = h \\ \{i_h \mid 0 \leqslant i_h \leqslant p-1\} & \text{if } l > h \end{cases} \quad \text{for } l, h = 1, \ldots, H_I \,. \tag{4.6}$$

Then subpartition $l = 1, \ldots, H_I$ of $\mathcal{R}_p$, denoted by $\mathcal{R}_{p,l}$, is given by

$$\mathcal{R}_{p,l} = \left\{ \mathbf{i} \in \mathcal{R}_p \mid (i_1, \ldots, i_{H_I}) \in \bigtimes_{h=1}^{H_I} \mathcal{S}_{p,l}^{(h)} \text{ and } (i_{H_I+1}, \ldots, i_H) \in \bar{\mathcal{R}} \right\} \,.$$

Finally,

$$\mathcal{R}_p = \bigcup_{l=1}^{H_I} \mathcal{R}_{p,l} \,, \quad \mathcal{R}_{p,l} \cap \mathcal{R}_{p,m} = \varnothing \quad \text{for } l \neq m \,.$$

Without loss of generality, the subpartitions $\mathcal{R}_{p,l}$ are assumed to be ordered within $\mathcal{R}_p$ according to increasing subpartition index, l.

The number of states within reachable state partitions $\mathcal{R}_{Low}$ through $\mathcal{R}_{High}$ is given by

$$n(Low, High) = \sum_{p=Low}^{High} \left|\bar{\mathcal{R}}\right| \left((p+1)^{H_I} - (p)^{H_I}\right) = \left|\bar{\mathcal{R}}\right| \left((High+1)^{H_I} - Low^{H_I}\right) \,.$$

Observe that the number of states in reachable state space partition $\mathcal{R}_p$ for $p \in \mathbb{Z}_{\geqslant 0}$ is $O(p^{H_I - 1})$.

In the metabolite synthesis model, we have

$$\begin{aligned} &\mathcal{S}_{p,1}^{(1)} = \{p\} \,, \quad \mathcal{S}_{p,1}^{(2)} = \mathcal{S}_{p,1}^{(3)} = \{0, \ldots, p\} \,, \\ &\mathcal{S}_{p,2}^{(1)} = \{0, \ldots, p-1\} \,, \quad \mathcal{S}_{p,2}^{(2)} = \{p\} \,, \quad \mathcal{S}_{p,2}^{(3)} = \{0, \ldots, p\} \,, \\ &\mathcal{S}_{p,3}^{(1)} = \mathcal{S}_{p,3}^{(2)} = \{0, \ldots, p-1\} \,, \quad \mathcal{S}_{p,3}^{(3)} = \{p\} \,, \end{aligned}$$

implying

$$\begin{aligned} &\bigtimes_{h=1}^{3} \mathcal{S}_{0,1}^{(h)} = \{(0,0,0)\} \,, \quad \bigtimes_{h=1}^{3} \mathcal{S}_{0,2}^{(h)} = \bigtimes_{h=1}^{3} \mathcal{S}_{0,3}^{(h)} = \varnothing \,, \\ &\bigtimes_{h=1}^{3} \mathcal{S}_{1,1}^{(h)} = \{(1,0,0), (1,0,1), (1,1,0), (1,1,1)\} \,, \\ &\bigtimes_{h=1}^{3} \mathcal{S}_{1,2}^{(h)} = \{(0,1,0), (0,1,1)\} \,, \quad \bigtimes_{h=1}^{3} \mathcal{S}_{1,3}^{(h)} = \{(0,0,1)\} \,, \\ &\bigtimes_{h=1}^{3} \mathcal{S}_{2,1}^{(h)} = \{(2,0,0), (2,0,1), (2,0,2), (2,1,0), (2,1,1), (2,1,2), \\ &\qquad\qquad (2,2,0), (2,2,1), (2,2,2)\} \,, \\ &\bigtimes_{h=1}^{3} \mathcal{S}_{2,2}^{(h)} = \{(0,2,0), (0,2,1), (0,2,2), (1,2,0), (1,2,1), (1,2,2)\} \,, \\ &\bigtimes_{h=1}^{3} \mathcal{S}_{2,3}^{(h)} = \{(0,0,2), (0,1,2), (1,0,2), (1,1,2)\} \,, \end{aligned}$$

and so on. Since

$$\bar{\mathcal{R}} = \{(0,0,0), \ldots, (1,1,1)\} ,$$

we obtain

$$\mathcal{R}_{p,1} = \left(\bigtimes_{h=1}^{3} \mathcal{S}_{p,1}^{(h)} \right) \times \bar{\mathcal{R}} = \{(p,0,0,0,0,0), \ldots, (p,p,p,1,1,1)\} ,$$

$$\mathcal{R}_{p,2} = \left(\bigtimes_{h=1}^{3} \mathcal{S}_{p,2}^{(h)} \right) \times \bar{\mathcal{R}} = \{(0,p,0,0,0,0), \ldots, (p-1,p,p,1,1,1)\} ,$$

$$\mathcal{R}_{p,3} = \left(\bigtimes_{h=1}^{3} \mathcal{S}_{p,3}^{(h)} \right) \times \bar{\mathcal{R}} = \{(0,0,p,0,0,0), \ldots, (p-1,p-1,p,1,1,1)\} .$$

Now, we are in a position to introduce the Kronecker representation of nonzero blocks in Q following the partitions of subset of states at each reachable state space partition. Nonzero blocks $Q(0,0)$, $Q(0,1)$, $Q(1,0)$, and $Q(p,w)$ for $p \in \mathbb{Z}_{\geqslant 0}$, $w = p-1, p, p+1$ are, respectively, (1×1), $(1 \times H_I)$, $(H_I \times 1)$, and $(H_I \times H_I)$ block matrices as in

$$Q(0,0) = \begin{pmatrix} Q(0,0)_{1,1} \end{pmatrix} , \quad Q(0,1) = \begin{pmatrix} Q(0,1)_{1,1} & \ldots & Q(0,1)_{1,H_I} \end{pmatrix} ,$$

$$Q(1,0) = \begin{pmatrix} Q(1,0)_{1,1} \\ \vdots \\ Q(1,0)_{H_I,1} \end{pmatrix} , \quad Q(p,w) = \begin{pmatrix} Q(p,w)_{1,1} & \ldots & Q(p,w)_{1,H_I} \\ \vdots & \ddots & \vdots \\ Q(p,w)_{H_I,1} & \ldots & Q(p,w)_{H_I,H_I} \end{pmatrix} .$$

Furthermore, blocks of $Q(p,w)$ can be written in terms of transition matrices and state-independent transition rates as in

$$Q(p,w)_{l,m} = \begin{cases} \tilde{Q}(p,w)_{l,m} - \operatorname{diag}\left(\sum_{w'=p-1}^{p+1} \sum_{l'=1}^{H_I} \tilde{Q}(p,w')_{l,l'} \mathbf{e} \right) \\ \qquad\qquad \text{if } l = m,\ p = w \\ \tilde{Q}(p,w)_{l,m} \quad \text{otherwise} \end{cases}$$

for $p \in \mathbb{Z}_{\geqslant 0}$, $w = p-1, p, p+1$, $l, m = 1, \ldots, H_I$, where

$$\tilde{Q}(p,w)_{l,m} = \sum_{k=1}^{K} \phi_k \left(\bigotimes_{h=1}^{H_I} Q_k^{(h)}(\mathcal{S}_{p,l}^{(h)}, \mathcal{S}_{w,m}^{(h)}) \right) \otimes \bar{Q}_k$$

and $Q_k^{(h)}(\mathcal{S}_{p,l}^{(h)}, \mathcal{S}_{w,m}^{(h)})$ denotes the submatrix of $Q_k^{(h)}$ incident on row indices in $\mathcal{S}_{p,l}^{(h)}$ and column indices in $\mathcal{S}_{w,m}^{(h)}$. The first summation in diag for $Q(p,w)_{l,m}$ should have a starting index of 0 rather than -1 for the equation of block $Q(0,0)_{1,1}$. Observe that beyond reachable state space partition $\mathcal{R}_0$, we have nonzero $(H_I \times H_I)$ block (sub)matrices along the subdiagonal, diagonal, and superdiagonal of Q.

In the metabolite synthesis model, $H_I = 3$; therefore, the nonzero blocks $Q(0,0)$, $Q(0,1)$, $Q(1,0)$, and $Q(p,w)$ for $p \in \mathbb{Z}_{\geqslant 0}$, $w = p-1, p, p+1$ (except

$Q(1,0)$) are, respectively, (1×1), (1×3), (3×1), and (3×3) block matrices. In particular, the seven blocks associated with $Q(0,0)$, $Q(0,1)$, $Q(1,0)$ are given by

$$\begin{aligned}
\tilde{Q}(0,0)_{1,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{0,1}^{(h)}, \mathcal{S}_{0,1}^{(h)}) \right) \otimes \bar{Q}_k = (0)_{8\times 8} \ , \\
\tilde{Q}(0,1)_{1,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{0,1}^{(h)}, \mathcal{S}_{1,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_1(1) \otimes (1,0) \otimes (1,0) \otimes \bar{Q}_1 + \beta_1(1) \otimes (1,0) \otimes (1,0) \otimes \bar{Q}_4 \ , \\
\tilde{Q}(0,1)_{1,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{0,1}^{(h)}, \mathcal{S}_{1,2}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_2(1) \otimes (1) \otimes (1,0) \otimes \bar{Q}_5 + \beta_1(1) \otimes (1) \otimes (1,0) \otimes \bar{Q}_8 \ , \\
\tilde{Q}(0,1)_{1,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{0,1}^{(h)}, \mathcal{S}_{1,3}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_3(1) \otimes (1) \otimes (1) \otimes \bar{Q}_9 + \beta_1(1) \otimes (1) \otimes (1) \otimes \bar{Q}_{12} \ , \\
\tilde{Q}(1,0)_{1,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{1,1}^{(h)}, \mathcal{S}_{0,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_1(1) \otimes (1,0)^T \otimes (1,0)^T \otimes \bar{Q}_2 + \beta_0(1) \otimes (1,0)^T \otimes (1,0)^T \otimes \bar{Q}_3 \ , \\
\tilde{Q}(1,0)_{2,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{1,2}^{(h)}, \mathcal{S}_{0,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_2(1) \otimes (1) \otimes (1,0)^T \otimes \bar{Q}_6 + \beta_0(1) \otimes (1) \otimes (1,0)^T \otimes \bar{Q}_7 \ , \\
\tilde{Q}(1,0)_{3,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{1,3}^{(h)}, \mathcal{S}_{0,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_3(1) \otimes (1) \otimes (1) \otimes \bar{Q}_{10} + \beta_0(1) \otimes (1) \otimes (1) \otimes \bar{Q}_{11} \ ,
\end{aligned}$$

the nine blocks associated with $Q(p,p-1)$ are given by

$$\begin{aligned}
\tilde{Q}(p,p-1)_{1,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p-1,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_1(p) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \bar{Q}_2 \\
&\quad + \beta_0(p) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \bar{Q}_3 \ , \\
\tilde{Q}(p,p-1)_{1,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p-1,2}^{(h)}) \right) \otimes \bar{Q}_k = 0_{8(p+1)^2\times 8(p-1)p} \ ,
\end{aligned}$$

$$
\begin{aligned}
\tilde{Q}(p,p-1)_{1,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p-1,3}^{(h)}) \right) \otimes \bar{Q}_k = 0_{8(p+1)^2 \times 8(p-1)^2} \ , \\
\tilde{Q}(p,p-1)_{2,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)}, \mathcal{S}_{p-1,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_2(\mathbf{e}_p)_{p\times 1} \otimes (p\mathbf{e}_p^T)_{1\times p} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \bar{Q}_6 \\
&\quad + \beta_0(\mathbf{e}_p)_{p\times 1} \otimes (p\mathbf{e}_p^T)_{1\times p} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \bar{Q}_7 \ , \\
\tilde{Q}(p,p-1)_{2,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)}, \mathcal{S}_{p-1,2}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_2 \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes (p) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \bar{Q}_6 \\
&\quad + \beta_0 \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes (p) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p} \otimes \bar{Q}_7 \ , \\
\tilde{Q}(p,p-1)_{2,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)}, \mathcal{S}_{p-1,3}^{(h)}) \right) \otimes \bar{Q}_k = 0_{8p(p+1)\times 8(p-1)^2} \ , \\
\tilde{Q}(p,p-1)_{3,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p-1,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_3(\mathbf{e}_p)_{p\times 1} \otimes \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (p\mathbf{e}_p^T)_{1\times p} \otimes \bar{Q}_{10} \\
&\quad + \beta_0(\mathbf{e}_p)_{p\times 1} \otimes \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (p\mathbf{e}_p^T)_{1\times p} \otimes \bar{Q}_{11} \ , \\
\tilde{Q}(p,p-1)_{3,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p-1,2}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_3 \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes (\mathbf{e}_p)_{p\times 1} \otimes (p\mathbf{e}_p^T)_{1\times p} \otimes \bar{Q}_{10} \\
&\quad + \beta_0 \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes (\mathbf{e}_p)_{p\times 1} \otimes (p\mathbf{e}_p^T)_{1\times p} \otimes \bar{Q}_{11} \ , \\
\tilde{Q}(p,p-1)_{3,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p-1,3}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \mu_3 \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes (p) \otimes \bar{Q}_{10} \\
&\quad + \beta_0 \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes \mathrm{diag}(\mathbf{e})_{p\times(p-1)} \otimes (p) \otimes \bar{Q}_{11} \ ,
\end{aligned}
$$

the nine blocks associated with $Q(p,p)$ are given by

$$
\begin{aligned}
\tilde{Q}(p,p)_{1,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_2(1) \otimes \mathrm{supdiag}(\mathbf{e})_{(p+1)\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)} \otimes \bar{Q}_5 \\
&\quad + \mu_2(1) \otimes \mathrm{subdiag}((1,\ldots,p)^T)_{(p+1)\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)} \otimes \bar{Q}_6 \\
&\quad + \beta_0(1) \otimes \mathrm{subdiag}((1,\ldots,p)^T)_{(p+1)\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)} \otimes \bar{Q}_7
\end{aligned}
$$

$$
\begin{aligned}
&+\beta_1(1)\otimes \mathrm{supdiag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_8\\
&+\lambda_3(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \mathrm{supdiag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_9\\
&+\mu_3(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \mathrm{subdiag}((1,\ldots,p)^T)_{(p+1)\times(p+1)}\otimes \bar{Q}_{10}\\
&+\beta_0(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \mathrm{subdiag}((1,\ldots,p)^T)_{(p+1)\times(p+1)}\otimes \bar{Q}_{11}\\
&+\beta_1(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \mathrm{supdiag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_{12}\ ,
\end{aligned}
$$

$$
\begin{aligned}
\tilde{Q}(p,p)_{1,2} &= \sum_{k=1}^{12}\phi_k\left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)},\mathcal{S}_{p,2}^{(h)})\right)\otimes \bar{Q}_k\\
&= \mu_1(p\mathbf{e}_p^T)_{1\times p}\otimes(\mathbf{e}_{p+1})_{(p+1)\times 1}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_2\\
&\quad+\beta_0(p\mathbf{e}_p^T)_{1\times p}\otimes(\mathbf{e}_{p+1})_{(p+1)\times 1}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_3\ ,\\
\tilde{Q}(p,p)_{1,3} &= \sum_{k=1}^{12}\phi_k\left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)},\mathcal{S}_{p,3}^{(h)})\right)\otimes \bar{Q}_k\\
&= \mu_1(p\mathbf{e}_p^T)_{1\times p}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p}\otimes(\mathbf{e}_{p+1})_{(p+1)\times 1}\otimes \bar{Q}_2\\
&\quad+\beta_0(p\mathbf{e}_p^T)_{1\times p}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times p}\otimes(\mathbf{e}_{p+1})_{(p+1)\times 1}\otimes \bar{Q}_3\ ,\\
\tilde{Q}(p,p)_{2,1} &= \sum_{k=1}^{12}\phi_k\left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)},\mathcal{S}_{p,1}^{(h)})\right)\otimes \bar{Q}_k\\
&= \lambda_1(\mathbf{e}_p)_{p\times 1}\otimes(\mathbf{e}_{p+1}^T)_{1\times(p+1)}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_1\\
&\quad+\beta_1(\mathbf{e}_p)_{p\times 1}\otimes(\mathbf{e}_{p+1}^T)_{1\times(p+1)}\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_4\ ,\\
\tilde{Q}(p,p)_{2,2} &= \sum_{k=1}^{12}\phi_k\left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)},\mathcal{S}_{p,2}^{(h)})\right)\otimes \bar{Q}_k\\
&= \lambda_1\mathrm{supdiag}(\mathbf{e})_{p\times p}\otimes(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_1\\
&\quad+\mu_1\mathrm{subdiag}((1,\ldots,p-1)^T)_{p\times p}\otimes(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_2\\
&\quad+\beta_0\mathrm{subdiag}((1,\ldots,p-1)^T)_{p\times p}\otimes(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_3\\
&\quad+\beta_1\mathrm{supdiag}(\mathbf{e})_{p\times p}\otimes(1)\otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_4\\
&\quad+\lambda_3\mathrm{diag}(\mathbf{e})_{p\times p}\otimes(1)\otimes \mathrm{supdiag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_9\\
&\quad+\mu_3\mathrm{diag}(\mathbf{e})_{p\times p}\otimes(1)\otimes \mathrm{subdiag}((1,\ldots,p)^T)_{(p+1)\times(p+1)}\otimes \bar{Q}_{10}\\
&\quad+\beta_0\mathrm{diag}(\mathbf{e})_{p\times p}\otimes(1)\otimes \mathrm{subdiag}((1,\ldots,p)^T)_{(p+1)\times(p+1)}\otimes \bar{Q}_{11}\\
&\quad+\beta_1\mathrm{diag}(\mathbf{e})_{p\times p}\otimes(1)\otimes \mathrm{supdiag}(\mathbf{e})_{(p+1)\times(p+1)}\otimes \bar{Q}_{12}\ ,\\
\tilde{Q}(p,p)_{2,3} &= \sum_{k=1}^{12}\phi_k\left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)},\mathcal{S}_{p,3}^{(h)})\right)\otimes \bar{Q}_k\\
&= \mu_2\mathrm{diag}(\mathbf{e})_{p\times p}\otimes(p\mathbf{e}_p^T)_{1\times p}\otimes(\mathbf{e}_{p+1})_{(p+1)\times 1}\otimes \bar{Q}_6\\
&\quad+\beta_0\mathrm{diag}(\mathbf{e})_{p\times p}\otimes(p\mathbf{e}_p^T)_{1\times p}\otimes(\mathbf{e}_{p+1})_{(p+1)\times 1}\otimes \bar{Q}_7\ ,
\end{aligned}
$$

$$
\begin{aligned}
\tilde{Q}(p,p)_{3,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_1 (\mathbf{e}_p)_{p\times 1} \otimes \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes \bar{Q}_1 \\
&\quad + \beta_1 (\mathbf{e}_p)_{p\times 1} \otimes \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes \bar{Q}_4 \ , \\
\tilde{Q}(p,p)_{3,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p,2}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_2 \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (\mathbf{e}_p)_{p\times 1} \otimes (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes \bar{Q}_5 \\
&\quad + \beta_1 \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (\mathbf{e}_p)_{p\times 1} \otimes (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes \bar{Q}_8 \ , \\
\tilde{Q}(p,p)_{3,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p,3}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_1 \mathrm{supdiag}(\mathbf{e})_{p\times p} \otimes \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (1) \otimes \bar{Q}_1 \\
&\quad + \mu_1 \mathrm{subdiag}((1,\ldots,p-1)^T)_{p\times p} \otimes \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (1) \otimes \bar{Q}_2 \\
&\quad + \beta_0 \mathrm{subdiag}((1,\ldots,p-1)^T)_{p\times p} \otimes \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (1) \otimes \bar{Q}_3 \\
&\quad + \beta_1 \mathrm{supdiag}(\mathbf{e})_{p\times p} \otimes \mathrm{diag}(\mathbf{e})_{p\times p} \otimes (1) \otimes \bar{Q}_4 \\
&\quad + \lambda_2 \mathrm{diag}(\mathbf{e})_{p\times p} \otimes \mathrm{supdiag}(\mathbf{e})_{p\times p} \otimes (1) \otimes \bar{Q}_5 \\
&\quad + \mu_2 \mathrm{diag}(\mathbf{e})_{p\times p} \otimes \mathrm{subdiag}((1,\ldots,p-1)^T)_{p\times p} \otimes (1) \otimes \bar{Q}_6 \\
&\quad + \beta_0 \mathrm{diag}(\mathbf{e})_{p\times p} \otimes \mathrm{subdiag}((1,\ldots,p-1)^T)_{p\times p} \otimes (1) \otimes \bar{Q}_7 \\
&\quad + \beta_1 \mathrm{diag}(\mathbf{e})_{p\times p} \otimes \mathrm{supdiag}(\mathbf{e})_{p\times p} \otimes (1) \otimes \bar{Q}_8 \ ,
\end{aligned}
$$

and the nine blocks associated with $Q(p,p+1)$ are given by

$$
\begin{aligned}
\tilde{Q}(p,p+1)_{1,1} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p+1,1}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_1 (1) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \bar{Q}_1 \\
&\quad + \beta_1 (1) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \bar{Q}_4 \ , \\
\tilde{Q}(p,p+1)_{1,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p+1,2}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_2 (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes (\mathbf{e}_{p+1})_{(p+1)\times 1} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \bar{Q}_5 \\
&\quad + \beta_1 (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes (\mathbf{e}_{p+1})_{(p+1)\times 1} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \bar{Q}_8 \ , \\
\tilde{Q}(p,p+1)_{1,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,1}^{(h)}, \mathcal{S}_{p+1,3}^{(h)}) \right) \otimes \bar{Q}_k \\
&= \lambda_3 (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)} \otimes (\mathbf{e}_{p+1})_{(p+1)\times 1} \otimes \bar{Q}_9 \\
&\quad + \beta_1 (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+1)} \otimes (\mathbf{e}_{p+1})_{(p+1)\times 1} \otimes \bar{Q}_{12} \ ,
\end{aligned}
$$

$$\tilde{Q}(p,p+1)_{2,1} = \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)}, \mathcal{S}_{p+1,1}^{(h)}) \right) \otimes \bar{Q}_k = 0_{8p(p+1)\times 8(p+2)^2} \ ,$$

$$\begin{aligned}\tilde{Q}(p,p+1)_{2,2} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)}, \mathcal{S}_{p+1,2}^{(h)}) \right) \otimes \bar{Q}_k \\ &= \lambda_2 \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (1) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \bar{Q}_5 \\ &\quad + \beta_1 \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (1) \otimes \mathrm{diag}(\mathbf{e})_{(p+1)\times(p+2)} \otimes \bar{Q}_8 \ ,\end{aligned}$$

$$\begin{aligned}\tilde{Q}(p,p+1)_{2,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,2}^{(h)}, \mathcal{S}_{p+1,3}^{(h)}) \right) \otimes \bar{Q}_k \\ &= \lambda_3 \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes (\mathbf{e}_{p+1})_{(p+1)\times 1} \otimes \bar{Q}_9 \\ &\quad + \beta_1 \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (\mathbf{e}_{p+1}^T)_{1\times(p+1)} \otimes (\mathbf{e}_{p+1})_{(p+1)\times 1} \otimes \bar{Q}_{12} \ ,\end{aligned}$$

$$\tilde{Q}(p,p+1)_{3,1} = \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p+1,1}^{(h)}) \right) \otimes \bar{Q}_k = 0_{8p^2\times 8(p+2)^2} \ ,$$

$$\tilde{Q}(p,p+1)_{3,2} = \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p+1,2}^{(h)}) \right) \otimes \bar{Q}_k = 0_{8p^2\times 8(p+1)(p+2)} \ ,$$

$$\begin{aligned}\tilde{Q}(p,p+1)_{3,3} &= \sum_{k=1}^{12} \phi_k \left(\bigotimes_{h=1}^{3} Q_k^{(h)}(\mathcal{S}_{p,3}^{(h)}, \mathcal{S}_{p+1,3}^{(h)}) \right) \otimes \bar{Q}_k \\ &= \lambda_3 \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (1) \otimes \bar{Q}_9 \\ &\quad + \beta_1 \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes \mathrm{diag}(\mathbf{e})_{p\times(p+1)} \otimes (1) \otimes \bar{Q}_{12} \ .\end{aligned}$$

Now, let us investigate the suitability of the squared Euclidean norm, that is, $g(i_1,i_2,i_3,i_4,i_5,i_6) = \sum_{h=1}^{6} i_h^2$, as the Lyapunov function for the set of parameters $\lambda_1 = \lambda_2 = \lambda_3 = 1.3$, $\mu_1 = \mu_2 = \mu_3 = 0.8$, $\beta_0 = 1$, and $\beta_1 = 0.5$. The corresponding drift from (4.3) is given by

$$\begin{aligned}d(\mathbf{i}) = &-3.6i_1^2 + 2i_1^2 i_4 - i_1 i_4 - 2.6 i_1 i_6 + 5.4 i_1 - 1.3 i_6 + 1.3 \\ &-3.6i_2^2 + 2i_2^2 i_5 - i_2 i_5 - 2.6 i_2 i_4 + 5.4 i_2 - 1.3 i_4 + 1.3 \\ &-3.6i_3^2 + 2i_3^2 i_6 - i_3 i_6 - 2.6 i_3 i_5 + 5.4 i_3 - 1.3 i_5 + 1.3 \ .\end{aligned}$$

Observe that $d(i_1,i_2,i_3,i_4,i_5,i_6)$ is a nonlinear function of six variables. The proof that $\mathcal{C}$ is finite follows from showing that there exists a finite superset of $\mathcal{C}$ [255]. The global maximum drift is computed as $\chi = 9.3$ using the HOM4PS2-2.0 package [215]. It is attained at state $(1,1,1,0,0,0)$. With a lower bound of 95% on the steady-state probability mass (i.e., $\varepsilon = 0.05$), we obtain $\gamma = 176.7$. This yields $(Low, High) = (0,12)$ with $n(0,12) = 17{,}576$ states.

In closing, we point out that various systems of stochastic chemical kinetics other than the metabolite synthesis model with repressilator have been modeled using Kronecker products. Gene expression [311], exclusive switch

[219], toggle switch [150, 217], metabolite synthesis with two metabolites, and one or two enzymes [297] are used as test cases in [18, 19, 107, 110]. Note that there are difficulties associated with analyzing systems of stochastic chemical kinetics that would be modeled as HMMs and SANs. Normally each countably infinite variable would be represented as a separate submodel with truncated state space. The control unit could still be represented as a grouped single submodel with finite state space. Hence, it is possible to consider a model with $(H_I + 1)$ submodels when $H_F > 0$ or H_I submodels when $H_F = 0$. With SANs [326], there would be K Kronecker product terms of which those that have state dependent $\alpha_k^{(h)}(i_h)$ values for $h = 1, \ldots, H$ would be represented with functional transitions in the corresponding factors. For instance, this implies all transition classes in Table 4.7. Among these transition classes, the most difficult ones to work with would be those that have dependencies on submodels associated with countably infinite variables (such as transition classes $2, 3, 6, 7, 10, 11$ in Table 4.7). When executing the vector–Kronecker product multiplication algorithm discussed in the next chapter, such transition classes would require a different evaluation for each value of the corresponding countably infinite variable, thereby complicating and slowing down the analysis process when the algorithm is employed. As for HMMs, transition classes that have dependencies on submodels associated with countably infinite variables would require a separate term for each state dependent value of the variable (and as many terms as there are state dependent values for the finite variable). Hence, each functional transition would need to be transformed to multiple Kronecker product terms. This not only would increase the number of Kronecker product terms by a number proportional to the sum of the truncated state space sizes of the countably infinite variables but would also make the corresponding factors in the Kronecker product terms extremely sparse, having a single nonzero entry. Therefore, any gain that would accrue from using a Kronecker-based representation would start to diminish.

In the next subsection, we discuss a call center model from [18] in which the interplay between finite variables and countably infinite variables is more intricate than that of the class of models considered in this subsection. This requires us to extend the form of the rates of transition classes in (4.4) so that the dependency on the values of variables is more general. In order to obtain a Kronecker representation for the nonzero blocks of Q, this extension necessitates the computation of partitions of state spaces of finite variables and subpartitions of state space partitions of countably infinite variables in (4.6).

4.4.2 A Call Center Model

Consider the parallel service system in Figure 4.2 known as the W-model for call centers under the fixed queue ratio routing control policy proposed in [171]. In this model, there are three types of customers, three types of infinite queues, and two types of server pools. Customers of type 1 can be in queue 1 or server pool 1, customers of type 3 can be in queue 3 or server pool 2, and customers of type 2 can be in queue 2 or either server pool. Since the server pools do not differentiate between the two types of customers they serve, $\mathbf{i} = (i_1, i_2, i_3, i_4, i_5)$ is a possible state representation, where i_1, i_2, and i_3 denote the number of customers in queues 1, 2, and 3, respectively, whereas i_4 and i_5 denote the number of busy servers in pools 1 and 2, respectively. Then $H = 5$, $H_I = 3$, and $H_F = 2$. Type h customers arrive to the model according to a Poisson process with rate λ_h for $h = 1, 2, 3$ and server pool h has S_h servers with exponentially distributed service times each working at a rate of μ_h for $h = 1, 2$. Hence, the submodel state spaces are given by

$$\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \mathbb{Z}_{\geqslant 0}, \quad \mathcal{S}_4 = \{0, \ldots, S_1\}, \quad \text{and} \quad \mathcal{S}_5 = \{0, \ldots, S_2\}.$$

Upon arrival, a type 1 customer joins pool 1 if it has idle servers (i.e., $i_4 < S_1$); otherwise (i.e., $i_4 = S_1$) it enters queue 1. Similarly, an arriving type 3 customer joins pool 2 if it has idle servers (i.e., $i_5 < S_2$); otherwise (i.e., $i_5 = S_2$) it enters queue 2. As for type 2 customers, upon arrival the customer joins the server pool with largest idleness imbalance if both pools have idle servers (i.e., $i_4 < S_1, i_5 < S_2$); otherwise (i.e., $i_4 < S_1, i_5 = S_2$ or $i_4 = S_1, i_5 < S_2$) it joins the pool which has an idle server. Idleness imbalance of a server pool is a function of the number of idle servers in that pool, the total number of idle servers, and the idleness imbalance ratio corresponding to the server pool. Letting the idleness imbalance ratios of server pools 1 and 2 be denoted by ω_4 and ω_5, their idleness imbalances are obtained as $(S_1 - i_4)/(S_1 + S_2 - i_4 - i_5) - \omega_4$ and $(S_2 - i_5)/(S_1 + S_2 - i_4 - i_5) - \omega_5$, respectively. If both pools of servers are busy (i.e., $i_4 = S_1$, $i_5 = S_2$), an arriving type 2 customer enters queue 2. Upon departure of a customer from a server pool, head of the queue with largest queueing imbalance among the queues feeding the pool joins it. The queueing imbalance of a queue is a function of the number of customers in that queue, total number of customers in all queues, and the queueing imbalance ratio corresponding to the queue. Letting the queueing imbalance ratios of queues 1, 2, and 3 be denoted by ω_1, ω_2, and ω_3, their queueing imbalances are obtained as $i_1/(i_1 + i_2 + i_3) - \omega_1$, $i_2/(i_1+i_2+i_3)-\omega_2$, and $i_3/(i_1+i_2+i_3)-\omega_3$, respectively. The transition classes of this model are given in Table 4.8 [18]. Note that $K = 19$, $S_1, S_2 \in \mathbb{Z}_{>0}$, λ_1, λ_2, λ_3, μ_1, $\mu_2 \in \mathbb{R}_{>0}$, and $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5) \in \mathbb{R}_{>0}^{1\times 5}$.

Observe that the transition classes of the call center model do not satisfy the separability condition of transition rates in (4.4). In fact, they are similar in form to functional transitions of SANs. Such transition rate functions can

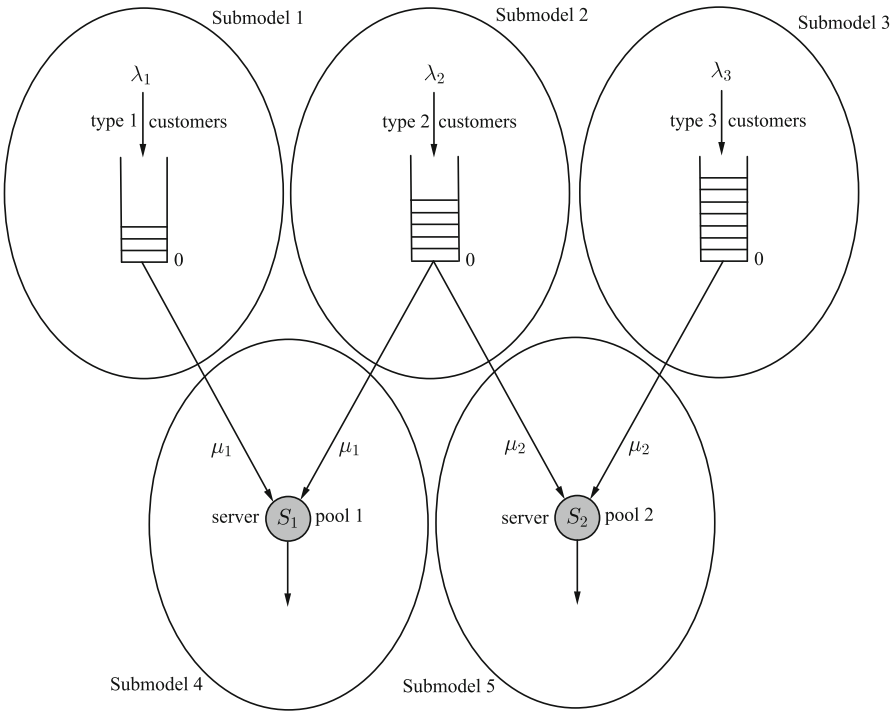


Fig. 4.2 Call center W–model with three types of customers and five submodels

pose a computational efficiency problem as mentioned before. However, due to the block tridiagonal nonzero structure of Q, it is possible to employ a matrix analytic solution method in which each nonzero block is processed once and thereby circumvent this problem [18, 19]. We return to this subject in Chapter 7.

When the separability condition in (4.4) is relaxed, transition matrices $Q_k^{(h)} \in \mathbb{R}_{\geqslant 0}^{|\mathcal{S}^{(h)}|\times|\mathcal{S}^{(h)}|}$ for $h = 1, \ldots, H$ and $k = 1, \ldots, K$ are given entrywise as

$$Q_k^{(h)}(i_h, j_h) = \begin{cases} 1 & \text{if } j_h = i_h + v_h^{(k)} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i_h, j_h \in \mathcal{S}^{(h)} \ .$$

Partitions $\mathcal{R}_p$ of reachable state space $\mathcal{R}$ and partitions $\mathcal{S}_{p,l}^{(h)}$ of submodel state spaces $\mathcal{S}^{(h)}$ for $l, h = 1, \ldots, H_I$ and $p \in \mathbb{Z}_{\geqslant 0}$ are still defined as in (4.5) and (4.6) with

$$\mathcal{S}_{p,l}^{(h)} = \mathcal{S}^{(h)} \quad \text{for } h = H_I + 1, \ldots, H,\ l = 1, \ldots, H_I$$

Table 4.8 Transition classes of W–model for call centers

k	$\alpha_k(\mathbf{i})$	$\mathbf{v}^{(k)}$
1	$\lambda_1 \ \mathbb{1}_{i_4=S_1}$	$\mathbf{e}_1^T$
2	$\lambda_1 \ \mathbb{1}_{i_4<S_1}$	$\mathbf{e}_4^T$
3	$\lambda_2 \ \mathbb{1}_{i_4=S_1} \ \mathbb{1}_{i_5=S_2}$	$\mathbf{e}_2^T$
4	$\lambda_2 \ \mathbb{1}_{i_4<S_1} \ \mathbb{1}_{i_5=S_2}$	$\mathbf{e}_4^T$
5	$\lambda_2 \ \mathbb{1}_{i_4=S_1} \ \mathbb{1}_{i_5<S_2}$	$\mathbf{e}_5^T$
6	$\lambda_2 \ \mathbb{1}_{i_4<S_1} \ \mathbb{1}_{i_5<S_2} \ \mathbb{1}_{(S_1-i_4)-(S_2-i_5)\geqslant(\omega_4-\omega_5)(S_1+S_2-i_4-i_5)}$	$\mathbf{e}_4^T$
7	$\lambda_2 \ \mathbb{1}_{i_4<S_1} \ \mathbb{1}_{i_5<S_2} \ \mathbb{1}_{(S_1-i_4)-(S_2-i_5)<(\omega_4-\omega_5)(S_1+S_2-i_4-i_5)}$	$\mathbf{e}_5^T$
8	$\lambda_3 \ \mathbb{1}_{i_5=S_2}$	$\mathbf{e}_3^T$
9	$\lambda_3 \ \mathbb{1}_{i_5<S_2}$	$\mathbf{e}_5^T$
10	$\mu_1 \ i_4 \ \mathbb{1}_{i_1=0} \ \mathbb{1}_{i_2=0}$	$-\mathbf{e}_4^T$
11	$\mu_1 \ i_4 \ \mathbb{1}_{i_1>0} \ \mathbb{1}_{i_2=0}$	$-\mathbf{e}_1^T$
12	$\mu_1 \ i_4 \ \mathbb{1}_{i_1=0} \ \mathbb{1}_{i_2>0}$	$-\mathbf{e}_2^T$
13	$\mu_1 \ i_4 \ \mathbb{1}_{i_1>0} \ \mathbb{1}_{i_2>0} \ \mathbb{1}_{(i_1-i_2)\geqslant(\omega_1-\omega_2)(i_1+i_2+i_3)}$	$-\mathbf{e}_1^T$
14	$\mu_1 \ i_4 \ \mathbb{1}_{i_1>0} \ \mathbb{1}_{i_2>0} \ \mathbb{1}_{(i_1-i_2)<(\omega_1-\omega_2)(i_1+i_2+i_3)}$	$-\mathbf{e}_2^T$
15	$\mu_2 \ i_5 \ \mathbb{1}_{i_2=0} \ \mathbb{1}_{i_3=0}$	$-\mathbf{e}_5^T$
16	$\mu_2 \ i_5 \ \mathbb{1}_{i_2>0} \ \mathbb{1}_{i_3=0}$	$-\mathbf{e}_2^T$
17	$\mu_2 \ i_5 \ \mathbb{1}_{i_2=0} \ \mathbb{1}_{i_3>0}$	$-\mathbf{e}_3^T$
18	$\mu_2 \ i_5 \ \mathbb{1}_{i_2>0} \ \mathbb{1}_{i_3>0} \ \mathbb{1}_{(i_2-i_3)\geqslant(\omega_2-\omega_3)(\omega_1+\omega_2+\omega_3)}$	$-\mathbf{e}_2^T$
19	$\mu_2 \ i_5 \ \mathbb{1}_{i_2>0} \ \mathbb{1}_{i_3>0} \ \mathbb{1}_{(i_2-i_3)<(\omega_2-\omega_3)(\omega_1+\omega_2+\omega_3)}$	$-\mathbf{e}_3^T$

for submodel state spaces corresponding to finite variables. Then $\mathcal{R}_{p,l}$ for $l = 1, \ldots, H_I$ satisfies

$$\mathcal{R}_{p,l} = \left\{ \mathbf{i} \in \mathcal{R}_p \mid (i_1, \ldots, i_H) \in \bigtimes_{h=1}^{H} \mathcal{S}_{p,l}^{(h)} \right\},$$

so that $\mathcal{R}_p = \bigcup_{p=1}^{H_I} \mathcal{R}_{p,l}$ and $\mathcal{R}_{p,l} \cap \mathcal{R}_{p,m} = \varnothing$ for $l \neq m$.

For the call center W-model, state space partitions are obtained from (4.6) and the above additional definition for finite state variables as

$$\begin{aligned}
&\mathcal{S}_{0,1}^{(1)} = \mathcal{S}_{0,1}^{(2)} = \mathcal{S}_{0,1}^{(3)} = \{0\}, \quad \mathcal{S}_{0,1}^{(4)} = \{0, \ldots, S_1\}, \quad \mathcal{S}_{0,1}^{(5)} = \{0, \ldots, S_2\}, \\
&\mathcal{S}_{p,1}^{(1)} = \{p\}, \quad \mathcal{S}_{p,1}^{(2)} = \mathcal{S}_{p,1}^{(3)} = \{0, \ldots, p\}, \quad \mathcal{S}_{p,1}^{(4)} = \{0, \ldots, S_1\}, \\
&\qquad \mathcal{S}_{p,1}^{(5)} = \{0, \ldots, S_2\}, \\
&\mathcal{S}_{p,2}^{(1)} = \{0, \ldots, p-1\}, \quad \mathcal{S}_{p,2}^{(2)} = \{p\}, \quad \mathcal{S}_{p,2}^{(3)} = \{0, \ldots, p\}, \quad \mathcal{S}_{p,2}^{(4)} = \{0, \ldots, S_1\}, \\
&\qquad \mathcal{S}_{p,2}^{(5)} = \{0, \ldots, S_2\}, \\
&\mathcal{S}_{p,3}^{(1)} = \mathcal{S}_{p,3}^{(2)} = \{0, \ldots, p-1\}, \quad \mathcal{S}_{p,3}^{(3)} = \{p\}, \quad \mathcal{S}_{p,3}^{(4)} = \{0, \ldots, S_1\}, \\
&\qquad \mathcal{S}_{p,3}^{(5)} = \{0, \ldots, S_2\} \quad \text{for } p > 0.
\end{aligned}$$

Note that partition $\mathcal{R}_{p,l}$ consists only of reachable states and is a subset of the Cartesian product of state space partitions of all variables since certain values of finite variables may yield unreachable states. Therefore, we next define subpartitions of $\mathcal{R}_{p,l}$ and $\mathcal{S}_{p,l}^{(h)}$ to eliminate any unreachable states due to finite variables.

Subpartition x of $\mathcal{R}_{p,l}$ denoted by $\mathcal{R}_{p,l,x}$ for $x = 1, \ldots, X_{p,l}$ is given by

$$\mathcal{R}_{p,l,x} = \left\{ \mathbf{i} \in \mathcal{R}_{p,l} \mid (i_1, \ldots, i_H) \in \bigtimes_{h=1}^{H} \mathcal{S}^{(h)}_{p,l,x_h} \right\},$$

so that $\mathcal{R}_{p,l} = \bigcup_{x=1}^{X_{p,l}} \mathcal{R}_{p,l,x}$, $\mathcal{R}_{p,l,x} \cap \mathcal{R}_{p,l,y} = \varnothing$ for $x \neq y$, where $\mathcal{S}^{(h)}_{p,l,x_h}$ is subpartition x_h of $\mathcal{S}^{(h)}_{p,l}$ for $h = 1, \ldots, H$, $l = 1, \ldots, H_I$, and $x_h = 1, \ldots, X^{(h)}_{p,l}$. Without loss of generality, reachable state space subpartitions $\mathcal{R}_{p,l,x}$ are assumed to be ordered within $\mathcal{R}_{p,l}$ lexicographically according to the corresponding subpartition indices, $(x_1, \ldots, x_H)$.

Note that subpartition $\mathcal{R}_{p,l,x}$ consists only of reachable states and is a subset of the Cartesian product of state space subpartitions of all variables since certain values of finite variables may yield unreachable states. We remark that the values of $X_{p,l}$ and $X^{(h)}_{p,l}$ for $h = 1, \ldots, H$ depend on the model. Although $X_{p,l} = \prod_{h=1}^{H} X^{(h)}_{p,l}$ is true for the metabolite synthesis model and other models of systems of stochastic chemical kinetics considered in [107] due to the fact that $X_{p,l} = 1$ and $X^{(h)}_{p,l} = 1$ for $h = 1, \ldots, H$, it need not be true in general.

In the W-model of call centers, a queue can be nonempty only if all servers capable of serving that queue are busy. Thus, $i_4 = S_1$ if $i_1 > 0$ or $i_2 > 0$, and $i_5 = S_2$ if $i_2 > 0$ or $i_3 > 0$. Due to these dependencies, subpartitions of $\mathcal{S}^{(h)}_{0,1}$ for $h = 1, \ldots, H$ and $\mathcal{R}_{0,1}$ can be written as

$$X^{(1)}_{0,1} = X^{(2)}_{0,1} = X^{(3)}_{0,1} = X^{(4)}_{0,1} = X^{(5)}_{0,1} = 1,$$

$$\mathcal{S}^{(1)}_{0,1,1} = \mathcal{S}^{(2)}_{0,1,1} = \mathcal{S}^{(3)}_{0,1,1} = \{0\},$$
$$\mathcal{S}^{(4)}_{0,1,1} = \{0, \ldots, S_1\}, \quad \mathcal{S}^{(5)}_{0,1,1} = \{0, \ldots, S_2\},$$

so that $X_{0,1} = 1$ and

$$\mathcal{R}_{0,1,1} = \{0\} \times \{0\} \times \{0\} \times \{0, \ldots, S_1\} \times \{0, \ldots, S_2\}.$$

Subpartitions of $\mathcal{S}^{(h)}_{p,1}$ for $h = 1, \ldots, H$ and $\mathcal{R}_{p,1}$ for $p > 0$ can be written as

$$X^{(1)}_{p,1} = 1, \quad X^{(2)}_{p,1} = X^{(3)}_{p,1} = X^{(4)}_{p,1} = X^{(5)}_{p,1} = 2,$$

$$\mathcal{S}^{(1)}_{p,1,1} = \{p\}, \quad \mathcal{S}^{(2)}_{p,1,1} = \{0\}, \quad \mathcal{S}^{(2)}_{p,1,2} = \{1, \ldots, p\},$$
$$\mathcal{S}^{(3)}_{p,1,1} = \{0\}, \quad \mathcal{S}^{(3)}_{p,1,2} = \{1, \ldots, p\}, \quad \mathcal{S}^{(4)}_{p,1,1} = \{0, \ldots, S_1 - 1\}, \quad \mathcal{S}^{(4)}_{p,1,2} = \{S_1\},$$
$$\mathcal{S}^{(5)}_{p,1,1} = \{0, \ldots, S_2 - 1\}, \quad \mathcal{S}^{(5)}_{p,1,2} = \{S_2\},$$

so that $X_{p,1} = 5$ and

$$\begin{aligned}
\mathcal{R}_{p,1,1} &= \{p\} \times \{0\} \times \{0\} \times \{S_1\} \times \{0, \ldots, S_2 - 1\}, \\
\mathcal{R}_{p,1,2} &= \{p\} \times \{0\} \times \{0\} \times \{S_1\} \times \{S_2\}, \\
\mathcal{R}_{p,1,3} &= \{p\} \times \{0\} \times \{1, \ldots, p\} \times \{S_1\} \times \{S_2\}, \\
\mathcal{R}_{p,1,4} &= \{p\} \times \{1, \ldots, p\} \times \{0\} \times \{S_1\} \times \{S_2\}, \\
\mathcal{R}_{p,1,5} &= \{p\} \times \{1, \ldots, p\} \times \{1, \ldots, p\} \times \{S_1\} \times \{S_2\}.
\end{aligned}$$

Subpartitions of $\mathcal{S}_{p,2}^{(h)}$ for $h = 1, \ldots, H$ and $\mathcal{R}_{p,2}$ for $p > 0$ can be written as

$$X_{p,2}^{(1)} = X_{p,2}^{(2)} = X_{p,2}^{(3)} = 1, \quad X_{p,2}^{(4)} = X_{p,2}^{(5)} = 2,$$

$$\begin{aligned}
&\mathcal{S}_{p,2,1}^{(1)} = \{0, \ldots, p-1\}, \quad \mathcal{S}_{p,2,1}^{(2)} = \{p\}, \quad \mathcal{S}_{p,2,1}^{(3)} = \{0, \ldots, p\}, \\
&\mathcal{S}_{p,2,1}^{(4)} = \{0, \ldots, S_1 - 1\}, \quad \mathcal{S}_{p,2,2}^{(4)} = \{S_1\}, \\
&\mathcal{S}_{p,2,1}^{(5)} = \{0, \ldots, S_2 - 1\}, \quad \mathcal{S}_{p,2,2}^{(5)} = \{S_2\},
\end{aligned}$$

so that $X_{p,2} = 1$ and

$$\mathcal{R}_{p,2,1} = \{0, \ldots, p-1\} \times \{p\} \times \{0, \ldots, p\} \times \{S_1\} \times \{S_2\}.$$

Subpartitions of $\mathcal{S}_{1,3}^{(h)}$ for $h = 1, \ldots, H$ and $\mathcal{R}_{1,3}$ can be written as

$$X_{1,3}^{(1)} = X_{1,3}^{(2)} = X_{1,3}^{(3)} = X_{1,3}^{(4)} = 1, \quad X_{1,3}^{(5)} = 2,$$

$$\begin{aligned}
&\mathcal{S}_{1,3,1}^{(1)} = \{0\}, \quad \mathcal{S}_{1,3,1}^{(2)} = \{0\}, \quad \mathcal{S}_{1,3,1}^{(3)} = \{1\}, \\
&\mathcal{S}_{1,3,1}^{(4)} = \{0, \ldots, S_1\}, \quad \mathcal{S}_{1,3,1}^{(5)} = \{0, \ldots, S_2 - 1\}, \quad \mathcal{S}_{1,3,2}^{(5)} = \{S_2\},
\end{aligned}$$

so that $X_{1,3} = 1$ and

$$\mathcal{R}_{1,3,1} = \{0\} \times \{0\} \times \{1\} \times \{0, \ldots, S_1\} \times \{S_2\}.$$

Furthermore, subpartitions of $\mathcal{S}_{p,3}^{(h)}$ for $h = 1, \ldots, H$ and $\mathcal{R}_{p,3}$ for $p > 1$ can be written as

$$X_{p,3}^{(1)} = X_{p,3}^{(2)} = 2, \quad X_{p,3}^{(3)} = 1, \quad X_{p,3}^{(4)} = 2, \quad X_{p,3}^{(5)} = 2,$$

$$\begin{aligned}
&\mathcal{S}_{p,3,1}^{(1)} = \{0\}, \quad \mathcal{S}_{p,3,2}^{(1)} = \{1, \ldots, p-1\}, \quad \mathcal{S}_{p,3,1}^{(2)} = \{0\}, \quad \mathcal{S}_{p,3,2}^{(2)} = \{1, \ldots, p-1\}, \\
&\mathcal{S}_{p,3,1}^{(3)} = \{p\}, \quad \mathcal{S}_{p,3,1}^{(4)} = \{0, \ldots, S_1 - 1\}, \quad \mathcal{S}_{p,3,2}^{(4)} = \{S_1\}, \\
&\mathcal{S}_{p,3,1}^{(5)} = \{0, \ldots, S_2 - 1\}, \quad \mathcal{S}_{p,3,2}^{(5)} = \{S_2\},
\end{aligned}$$

so that $X_{p,3} = 5$ and

$$\begin{aligned}
\mathcal{R}_{p,3,1} &= \{0\} \times \{0\} \times \{p\} \times \{0, \ldots, S_1 - 1\} \times \{S_2\}, \\
\mathcal{R}_{p,3,2} &= \{0\} \times \{0\} \times \{p\} \times \{S_1\} \times \{S_2\}, \\
\mathcal{R}_{p,3,3} &= \{0\} \times \{1, \ldots, p-1\} \times \{p\} \times \{S_1\} \times \{S_2\}, \\
\mathcal{R}_{p,3,4} &= \{1, \ldots, p-1\} \times \{0\} \times \{p\} \times \{S_1\} \times \{S_2\}, \\
\mathcal{R}_{p,3,5} &= \{1, \ldots, p-1\} \times \{1, \ldots, p-1\} \times \{p\} \times \{S_1\} \times \{S_2\}.
\end{aligned}$$

Now, we generalize the Kronecker representation of nonzero blocks in Q given in the previous subsection following the subpartitions of reachable state space partitions $\mathcal{R}_p$. The nonzero blocks $Q(0,0)$, $Q(0,1)$, $Q(1,0)$, and $Q(p,w)$ for $p > 0$, $w = p-1, p, p+1$ are, respectively, (1×1), $(1 \times H_I)$, $(H_I \times 1)$, and $(H_I \times H_I)$ block matrices as in

$$Q(0,0) = \left(Q(0,0)_{1,1} \right), \quad Q(0,1) = \left(Q(0,1)_{1,1} \ \ldots \ Q(0,1)_{1,H_I} \right),$$

$$Q(1,0) = \begin{pmatrix} Q(1,0)_{1,1} \\ \vdots \\ Q(1,0)_{H_I,1} \end{pmatrix}, \quad Q(p,w) = \begin{pmatrix} Q(p,w)_{1,1} & \ldots & Q(p,w)_{1,H_I} \\ \vdots & \ddots & \vdots \\ Q(p,w)_{H_I,1} & \ldots & Q(p,w)_{H_I,H_I} \end{pmatrix},$$

where $Q(p,w)_{l,m}$ is an $(X_{p,l} \times X_{w,m})$ block matrix given by

$$Q(p,w)_{l,m} = \begin{pmatrix} Q(p,w)_{(l,1),(m,1)} & \ldots & Q(p,w)_{(l,1),(m,X_{w,m})} \\ \vdots & \ddots & \vdots \\ Q(p,w)_{(l,X_{p,l}),(m,1)} & \ldots & Q(p,w)_{(l,X_{p,l}),(m,X_{w,m})} \end{pmatrix}.$$

Furthermore, blocks of $Q(p,w)_{l,m}$ can be written in terms of transition rates and transition matrices as in

$$Q(p,w)_{(l,x),(m,y)} = \begin{cases} \tilde{Q}(p,w)_{(l,x),(m,y)} - \tilde{Q}_D(p,w)_{(l,x),(m,y)} & \text{if } \ l = m, \ p = w \\ \tilde{Q}(p,w)_{(l,x),(m,y)} & \text{otherwise} \end{cases}$$

for $p \in \mathbb{Z}_{\geqslant 0}$, $w = p-1, p, p+1$, $l, m = 1, \ldots, H_I$, and $x = 1, \ldots, X_{p,l}$, $y = 1, \ldots, X_{w,m}$, where

$$\tilde{Q}_D(p,w)_{(l,x),(m,y)} = \operatorname{diag}\left(\sum_{w'=p-1}^{p+1} \sum_{l'=1}^{H_I} \sum_{y'=1}^{X_{w,m}} \tilde{Q}(p,w')_{(l,x),(l',y')} \mathbf{e} \right),$$

$$\tilde{Q}(p,w)_{(l,x),(m,y)} = \textstyle\sum_{k=1}^{K} \alpha_k(\mathbf{i}) \left(\bigotimes_{h=1}^{H} Q_k^{(h)}(\mathcal{S}_{p,l,x_h}^{(h)}, \mathcal{S}_{w,m,y_h}^{(h)}) \right),$$

$\alpha_k(\mathbf{i})$ is the transition rate computed at state $\mathbf{i} \in \times_{h=1}^{H} \mathcal{S}_{p,l,x_h}^{(h)}$ and $Q_k^{(h)}(\mathcal{S}_{p,l,x_h}^{(h)}, \mathcal{S}_{w,m,y_h}^{(h)})$ denotes the submatrix of $Q_k^{(h)}$ incident on row indices in $\mathcal{S}_{p,l,x_h}^{(h)}$ and column indices in $\mathcal{S}_{w,m,y_h}^{(h)}$. The first summation in diag should

have a starting index of 0 rather than -1 for the equation of the blocks $Q(0,0)_{(1,1),(1,1)},\ldots,Q(0,0)_{(1,X_{0,1}),(1,X_{0,1})}$, and the second summation in diag should have an ending index of 1 rather than H_I for the equation of the blocks $Q(1,1)_{(1,1),(1,1)},\ldots,Q(1,1)_{(H_I,X_{1,H_I}),(H_I,X_{1,H_I})}$ when $w'=p-1$.

The nonzero blocks are generally very sparse and have nonzero entries that may depend on the reachable state space partition index. Clearly, the ordering of states within a reachable state space partition is only fixed up to a permutation. Observe that transitions are possible only between adjacent reachable state space partitions and the number of states within each partition increases with increasing index number. The latter is due to the increase in the number of different possibilities for the H_I countably infinite variables according to the reachable state space partition index definition in (4.5).

For call center models, Lyapunov functions should be carefully chosen by considering the transitions of the models since dependencies among their variables are more intricate. We let $\lambda_1=15$, $\lambda_2=16$, $\lambda_3=13$, $\mu_1=5$, $\mu_2=7$, $S_1=9$, $S_2=6$, and $\omega=(0.3,0.8,0.4,0.7,0.5)$ be the parameters. Then the drift from (4.3) is given by

$$\begin{aligned} d(\mathbf{i}) = \quad & (2(i_1+i_2+i_4-4.6)+1)\,(15+16\ \mathbb{1}_{i_5=6}+16\ \mathbb{1}_{i_4<9}\ \mathbb{1}_{i_5<6}\ \mathbb{1}_{3i_5\geqslant 2i_4}) \\ & +(2(i_2+i_3+i_5-3)+1)\,(13+16\ \mathbb{1}_{i_4=9}+16\ \mathbb{1}_{i_4<9}\ \mathbb{1}_{i_5<6}\ \mathbb{1}_{3i_5<2i_4}) \\ & +(-2(i_1+i_2+i_4-4.6)+1) \\ & \quad (5i_4+7i_5\ \mathbb{1}_{i_2>0}(\mathbb{1}_{i_3=0}+\mathbb{1}_{i_3>0}\ \mathbb{1}_{-2i_1+3i_2-7i_3\geqslant 0})) \\ & +(-2(i_2+i_3+i_5-3)+1) \\ & \quad (7i_5+5i_4\ \mathbb{1}_{i_2>0}(\mathbb{1}_{i_1=0}+\mathbb{1}_{i_1>0}\ \mathbb{1}_{3i_1-i_2+i_3<0})) \end{aligned}$$

for the Lyapunov function

$$g(\mathbf{i})=(i_1+i_2+i_4-r_1)^2+(i_2+i_3+i_5-r_2)^2,$$

where $r_1=(2\lambda_1+\lambda_2)/(2\mu_1)$ and $r_2=(2\lambda_3+\lambda_2)/(2\mu_2)$ are used to obtain a tighter upper bound. Note that r_1 is an approximation to the average number of busy servers in pool 1. Hence, $(i_1+i_2+i_4-r_1)$ is an approximation to the total number of customers in queues 1 and 2 plus the average number of idle servers in pool 1 if $i_4=S_1$. It is an approximation to the number of customers in queue 2 plus the difference between the number of busy servers and the average number of busy servers in pool 1 if $i_4<S_1$ (i.e., $i_1=0$). The explanation for r_2 is similar. In this way, we have a Lyapunov function that depends on the real parameters of the model. Since the drift for this model is not a nonlinear function of the countably infinite variables, there is no need to resort to the HOM4PS2-2.0 package. The global maximum drift is computed as $\chi=82$. For $\varepsilon=0.1$, we obtain $\gamma=738$ which yields $(Low,High)=(0,26)$ with $n(0,26)=20{,}142$ states.

We remark that various parallel service models other than the W-model for call centers have been represented using Kronecker products. The N-model

with two types of customers under the threshold routing policy proposed in [23] and the V-model with 2, 3, and 4 types of customers under the static priority control policy proposed in [170] are used as test cases in [18, 19, 110, 111]. In the next subsection, we consider another problem from queueing systems which also has countably infinite state variables. However, it will be represented using Kronecker products differently compared to the models in this and the previous subsections.

4.4.3 A Retrial Queueing Model

Retrial queues arise in various application areas such as call centers [262, 282], computer networks, and telecommunication systems [10, 79, 207]. They are models of systems in which a customer upon arrival finding all servers busy joins an infinite *retrial queue* (or *orbit*) and retries to receive service later [8, 9, 11, 131, 159, 198]. In Kendall notation [196], we consider the multiclass MAP/PH/S queueing model in Figure 4.3 with acyclic PH [34] retrials that is the subject of [112]. This is an S-server retrial queueing model which admits C classes of customers following *Markovian arrival processes* (MAPs) [250], *phase–type* (PH) [251] distributed service times, and acyclic PH distributed retrial times. In this model, when an arriving customer of class c finds all servers busy, it joins orbit c and retries to capture a server after a random amount of time for $c = 1, \ldots, C$. If a retrial customer in orbit c attempts to receive service when there are no idle servers, it is blocked and retries later.

The retrial queueing model in [112] has $S \geqslant 1$ homogeneous servers and $C \geqslant 1$ customer classes. Since the retrial queue represented by the orbits is infinite, we associate the first H_I submodels with the C orbits. Submodels H_I+1 through H_I+C are associated with MAPs of customers, and submodels $H_I + C + 1$ through $H_I + H_F$ are associated with PH service distributions of customers (see Figure 4.3). The H_F submodels associated with MAPs and PH services have finite state spaces. Hence, we again have $H = H_I + H_F$. Now, we explain the distributions associated with interarrival times, service times, and retrial times in more detail.

A MAP may be viewed as an irreducible CTMC with some marked transitions characterizing arrivals as in [71]. In this interpretation, a MAP with representation (B_0,B_1) of order $m \in \mathbb{Z}_{>0}$ is an irreducible CTMC with state space $\{0, \ldots, m-1\}$ and irreducible generator matrix $B_0 + B_1$, where $B_0 \in \mathbb{R}^{m \times m}$ is a nonsingular matrix with negative diagonal and nonnegative off-diagonal entries and $B_1 \in \mathbb{R}_{\geqslant 0}^{m \times m}$. The MAP representation describes a stochastic process in which B_0 represents transitions among states without an arrival, whereas B_1 represents transitions among states upon one customer arrival. States of the MAP are named *phases*.

A PH distribution with representation $(\boldsymbol{\beta},T)$ of order $m \in \mathbb{Z}_{>0}$ is the distribution of *time until absorption* in state m of a CTMC with state space

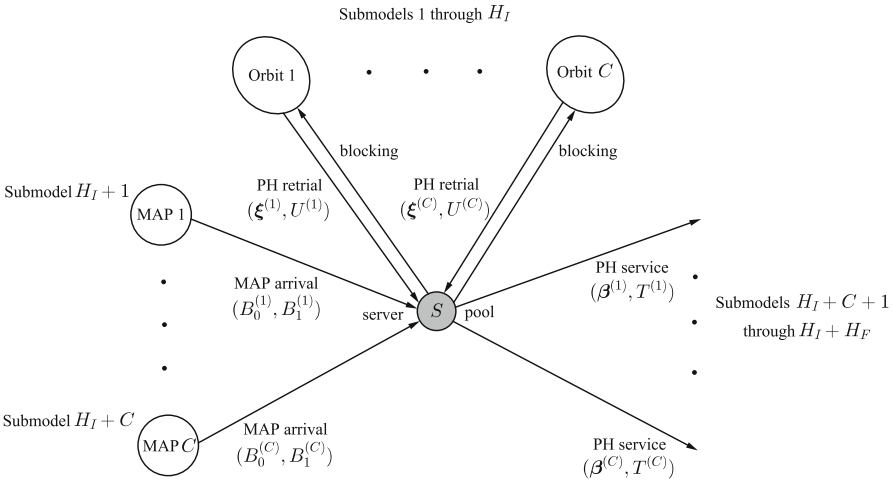


Fig. 4.3 Multiclass MAP/PH/S queueing model with acyclic PH retrials

$\{0, \ldots, m\}$, generator matrix

$$\hat{T} = \begin{pmatrix} T & \mathbf{T}_0 \\ \mathbf{0} & 0 \end{pmatrix} \in \mathbb{R}^{(m+1)\times(m+1)},$$

and initial probability vector $(\boldsymbol{\beta}, 1 - \boldsymbol{\beta}\mathbf{e}) \in \mathbb{R}_{\geqslant 0}^{1\times(m+1)}$, where $\boldsymbol{\beta} \in \mathbb{R}_{\geqslant 0}^{1\times m}$ and $T \in \mathbb{R}^{m\times m}$ is a nonsingular matrix with negative diagonal and nonnegative off-diagonal entries. States $\{0, \ldots, m-1\}$ in $\hat{T}$ are *transient* and said to be phases of the PH distribution. We assume that the process starts in one of the transient states after each absorption, that is, $\boldsymbol{\beta}\mathbf{e} = 1$. Since $\hat{T}$ is the generator matrix of a CTMC, we have $\mathbf{T}_0 = -T\mathbf{e} \in \mathbb{R}_{\geqslant 0}^{m\times 1}$. Accurate computation of moments of time until absorption for the PH distribution is addressed in [99, 102].

A PH distribution with representation ($\boldsymbol{\xi}$,U) is called *acyclic* if its states can be ordered in such a way that U is an upper-triangular matrix. Acyclicity of the PH distribution for retrials is needed in the sufficiency proof for the existence of steady-state with the particular Lyapunov function chosen [112]. Acyclic PH distributions are a subclass of PH distributions but are considered to be as powerful as them since both classes are dense in the set of nonnegative distributions (see, for instance, [71]). Therefore the model we consider is quite general.

Customers of class c arrive according to a MAP with representation $(B_0^{(c)}, B_1^{(c)})$ of order $m_{arv}^{(c)}$. Since $B_0^{(c)} + B_1^{(c)}$ is by definition irreducible, its steady-state vector denoted by $\boldsymbol{\theta}^{(c)} \in \mathbb{R}_{\geqslant 0}^{1\times m_{arv}^{(c)}}$ exists and satisfies

$$\boldsymbol{\theta}^{(c)}(B_0^{(c)} + B_1^{(c)}) = \mathbf{0}, \quad \boldsymbol{\theta}^{(c)}\mathbf{e} = 1.$$

Steady-state vector $\boldsymbol{\theta}^{(c)}$ of the MAP is used in defining the *average arrival rate* of class c customers as

$$\lambda^{(c)} = \boldsymbol{\theta}^{(c)} B_1^{(c)} \mathbf{e}.$$

The service time of class c customers follows a PH distribution with representation $(\boldsymbol{\beta}^{(c)}, T^{(c)})$ of order $m_{srv}^{(c)}$ and $\mathbf{T}_0^{(c)} = -T^{(c)}\mathbf{e}$. The *average service rate* of class c customers is given by

$$\mu^{(c)} = \frac{1}{\boldsymbol{\beta}^{(c)}(-T^{(c)})^{-1}\mathbf{e}}.$$

The retrial time of class c customers follows an acyclic PH distribution with representation $(\boldsymbol{\xi}^{(c)}, U^{(c)})$ of order $m_{rtl}^{(c)}$ and $\mathbf{U}_0^{(c)} = -U^{(c)}\mathbf{e}$. Without loss of generality, we assume that $U^{(c)}$ is upper-triangular. The *average retrial rate* of class c customers is given by

$$\delta^{(c)} = \frac{1}{\boldsymbol{\xi}^{(c)}(-U^{(c)})^{-1}\mathbf{e}}.$$

We remark that $-T^{(c)}$ and $-U^{(c)}$ are nonsingular M-matrices [242] by their definitions. Therefore, in the denominators of the expressions for average service rate and average retrial rate of class c customers, we have that $(-T^{(c)})^{-1} \geqslant 0$ and $(-U^{(c)})^{-1} \geqslant 0$. Each of these nonnegative matrices gets pre-multiplied with a probability vector and post-multiplied with a vector of 1's. By using arguments based on the nonzero structure of the inverse of a nonsymmetric matrix in [155], it is possible to show that the denominators in both expressions are positive, and so are the rates.

Example 6. Now let us consider a C-class MAP/PH/S queueing model with acyclic PH retrials that has the parameters

$$C = S = 2, \quad m_{arv}^{(1)} = m_{srv}^{(1)} = m_{rtl}^{(1)} = 2, \quad \text{and} \quad m_{arv}^{(2)} = m_{srv}^{(2)} = m_{rtl}^{(2)} = 1$$

in which the vectors and matrices describing the arrivals, services, and retrials are given by

$$B_0^{(1)} = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}, \quad B_1^{(1)} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad B_0^{(2)} = (-0.4), \quad B_1^{(2)} = (0.4),$$

$$\boldsymbol{\beta}^{(1)} = (0.5, 0.5), \quad T^{(1)} = \begin{pmatrix} -1.25 & 0.25 \\ 0.25 & -3.25 \end{pmatrix}, \quad \boldsymbol{\beta}^{(2)} = (1), \quad T^{(2)} = (-0.5),$$

$$\boldsymbol{\xi}^{(1)} = (1, 0), \quad U^{(1)} = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}, \quad \boldsymbol{\xi}^{(2)} = (1), \quad U^{(2)} = (-0.5).$$

Observe that class 1 customers arrive with a rate of 1 only if their MAP is in phase 1, and when the arrival takes place, the MAP makes a transition to

phase 0 as indicated by entry $B_1^{(1)}(1,0)$. On the other hand, the inter-arrival times of class 2 customers are exponentially distributed with rate 0.4 since their MAP has a single phase for which $B_1^{(2)}(0,0) = 0.4$. We remark that $U^{(1)}$ and $U^{(2)}$ are both upper-triangular, thereby corresponding to transition matrices associated with acyclic PH distributions.

The vectors $\mathbf{T}_0^{(c)}$ and $\mathbf{U}_0^{(c)}$ of PH services and acyclic PH retrials are obtained for $c = 1, 2$ as

$$\mathbf{T}_0^{(1)} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad \mathbf{T}_0^{(2)} = (0.5), \quad \mathbf{U}_0^{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \mathbf{U}_0^{(2)} = (0.5).$$

Since, $\boldsymbol{\theta}^{(1)} = (0.5, 0.5)$ and $\boldsymbol{\theta}^{(2)} = (1)$, we have $\lambda^{(1)} = \lambda^{(2)} = 0.5$ for the average arrival rates of class 1 and class 2 customers. Regarding the average service and average retrial rates, we obtain $\mu^{(1)} = 1.6$, $\mu^{(2)} = 0.5$, and $\delta^{(1)} = \delta^{(2)} = 0.5$.

Modeling and analysis of a system with arrival and service processes having more than one exponentially distributed phase is relatively complicated due to having to represent phases of arrival and service processes. Therefore, a single-class MAP/PH/S queue with PH retrials is studied in [75] using simulation. Modeling and analysis of a multiclass retrial queue in which all customers can join an orbit is even more difficult since its joint queue length process is a random walk on the multidimensional integer lattice [11]. In our model, arrival and service processes are multidimensional due to multiple classes of customers, but finite along the corresponding dimensions since the number of arrival phases, number of servers and service phases are finite. Hence, they can be handled systematically within the Kronecker setting as before. However, when the queueing system has PH retrials with a finite number of phases for each class of customers but a potentially infinite capacity in each orbit, the state vector which needs to represent the number of customers in all retrial phases becomes countably infinite along the dimensions associated with retrials. This results in a random walk on the multidimensional integer lattice with one or more countably infinite dimensions, and therefore, necessitates truncation of the countably infinite state space for analysis purposes. We remark that this was also the case for the metabolite synthesis and call center models considered in the previous two subsections.

A single-class MAP/PH/S queue with PH retrials is modeled in [75] using a multidimensional CTMC. We model its C-class counterpart similarly and represent the states of the CTMC with the multidimensional vector

$$\mathbf{i} = (i_1, \ldots, i_{H_I+H_F}) \in \mathcal{R} \quad \text{with} \quad H_I = m_{rtl} \quad \text{and} \quad H_F = m_{arv} + m_{srv},$$

where H_I is the number of dimensions with countably infinite state spaces, H_F is the number of dimensions with finite state spaces, $\mathcal{R}$ is the reachable state space, m_{rtl}, m_{arv}, m_{srv} are, respectively, the number of dimensions allocated to retrials, arrivals, and services. Here,

$$m_{rtl} = \sum_{c=1}^{C} m_{rtl}^{(c)}, \quad m_{arv} = C, \quad m_{srv} = \sum_{c=1}^{C} m_{srv}^{(c)}.$$

That is, we represent the number of customers of a particular class at a given phase in retrial or service as a separate dimension, whereas each class of customers is assigned a separate dimension for arrivals. Note that another representation could be to keep the phase of each server in a single but different dimension of the state. However, as discussed in [197, 273], this alternative approach results in a larger reachable state space $\mathcal{R}$, and therefore, is not preferred.

For the particular ordering of submodels considered, the multidimensional CTMC has the reachable state space

$$\mathcal{R} = \mathcal{S}_{rtl} \times \mathcal{S}_{arv} \times \mathcal{R}_{srv},$$

where

$$\mathcal{S}_{rtl} = \mathbb{Z}_{\geqslant 0}^{m_{rtl}}, \quad \mathcal{S}_{arv} = \bigtimes_{c=1}^{C} \left\{0, \ldots, m_{arv}^{(c)} - 1\right\}, \quad \mathcal{S}_{srv} = \bigtimes_{o=1}^{m_{srv}} \{0, \ldots, S\},$$

and $\mathcal{R}_{srv} = \{(i_{m_{rtl}+m_{arv}+1}, \ldots, i_H) \in \mathcal{S}_{srv} \mid (i_{m_{rtl}+m_{arv}+1}, \ldots, i_H)\mathbf{e} \leqslant S\}$.

The upper bound, S, on the sum of the values of state variables associated with services, that is, the total number of busy servers, ensures that $\mathcal{S}_{srv}$ and $\mathcal{R}_{srv}$ are finite. In fact,

$$|\mathcal{R}_{srv}| = \sum_{s=0}^{S} \frac{(s + m_{srv} - 1)!}{s!\ (m_{srv} - 1)!}$$

since $|\mathcal{R}_{srv}|$ is the total number of ways in which s busy servers can be distributed among m_{srv} different phases for $s = 0, \ldots, S$.

Now, let us introduce the base indices of state variables associated with retrial, arrival, and service dimensions as

$$b_{rtl}^{(c)} = 1 + \sum_{c'=1}^{c-1} m_{rtl}^{(c')}, \quad b_{arv} = 1 + m_{rtl}, \quad b_{srv}^{(c)} = 1 + m_{rtl} + m_{arv} + \sum_{c'=1}^{c-1} m_{srv}^{(c')}$$

for customers of class c. We also let $s(\mathbf{i})$ denote the number of busy servers in state $\mathbf{i}$, so that

$$s(\mathbf{i}) = \sum_{c=1}^{C} \sum_{o=0}^{m_{srv}^{(c)}-1} i_{b_{srv}^{(c)}+o}.$$

We will be using base indices and number of busy servers in specifying transitions and providing suitable Lyapunov functions for the model.

Example 6. (cntd.) The chosen parameters yield

$$m_{rtl} = 3, \quad m_{arv} = 2, \quad \text{and} \quad m_{srv} = 3,$$

which suggest $H_I = 3$, $H_F = 5$, $H = 8$, that is, an 8-dimensional model with reachable state space $\mathcal{R} = \mathcal{S}_{rtl} \times \mathcal{S}_{arv} \times \mathcal{R}_{srv}$ with

$$\mathcal{S}_{rtl} = \mathbb{Z}^3_{\geqslant 0}, \quad \mathcal{S}_{arv} = \{0,1\} \times \{0\}, \quad \text{and}$$

$$\begin{aligned}\mathcal{R}_{srv} = \{&(0,0,0),(0,0,1),(0,0,2),(0,1,0),(0,1,1),(0,2,0),\\ &(1,0,0),(1,0,1),(1,1,0),(2,0,0)\}.\end{aligned}$$

For instance, state $(0,0,1)$ in $\mathcal{R}_{srv}$ corresponds to one busy server in phase 0 serving one class 2 customer, while state $(0,2,0)$ corresponds to two busy servers both in phase 1 serving a total of two class 1 customers. Furthermore,

$$b^{(1)}_{rtl} = 1, \quad b^{(2)}_{rtl} = 3, \quad b_{arv} = 4, \quad b^{(1)}_{srv} = 6, \quad b^{(2)}_{srv} = 8,$$

and we have

$$s(\mathbf{i}) = i_6 + i_7 + i_8.$$

In Table 4.9, we provide the seven transition classes of the multiclass MAP/PH/S queueing model with acyclic PH retrials for class c customers, $c = 1, \ldots, C$. For C classes, there are altogether $K = 7C$ transition classes. For each class of customers, the first three transitions are due to MAP arrivals. The first one corresponds to a local transition in submodel $b_{arv} + c - 1$ with a phase change from $i_{b_{arv}+c-1}$ to o at rate $B^{(c)}_0(i_{b_{arv}+c-1}, o)$. The second one corresponds to a synchronizing transition involving submodels $b_{arv} + c - 1$ and $b^{(c)}_{rtl} + o'$ with an arrival in phase $i_{b_{arv}+c-1}$ and making a transition to phase o at rate $B^{(c)}_1(i_{b_{arv}+c-1}, o)$, joining orbit c and starting the acyclic PH

Table 4.9 Transition classes of the multiclass MAP/PH/S queueing model with acyclic PH retrials for class c customers, $c = 1, \ldots, C$

k	$\alpha_k(\mathbf{i})$	$\mathbf{v}^{(k)}$
$7(c-1)+1$	$B^{(c)}_0(i_{b_{arv}+c-1}, o)\ \mathbb{1}_{o \neq i_{b_{arv}+c-1}}$	$((o - i_{b_{arv}+c-1})\mathbf{e}_{b_{arv}+c-1})^T$
$7(c-1)+2$	$B^{(c)}_1(i_{b_{arv}+c-1}, o)\ \boldsymbol{\xi}^{(c)}(o')\ \mathbb{1}_{s(\mathbf{i})=S}$	$((o - i_{b_{arv}+c-1})\mathbf{e}_{b_{arv}+c-1} + \mathbf{e}_{b^{(c)}_{rtl}+o'})^T$
$7(c-1)+3$	$B^{(c)}_1(i_{b_{arv}+c-1}, o)\ \boldsymbol{\beta}^{(c)}(o')\ \mathbb{1}_{s(\mathbf{i})<S}$	$((o - i_{b_{arv}+c-1})\mathbf{e}_{b_{arv}+c-1} + \mathbf{e}_{b^{(c)}_{srv}+o'})^T$
$7(c-1)+4$	$i_{b^{(c)}_{rtl}+o}\ U^{(c)}(o, o')\ \mathbb{1}_{o \neq o'}$	$(\mathbf{e}_{b^{(c)}_{rtl}+o'} - \mathbf{e}_{b^{(c)}_{rtl}+o})^T$
$7(c-1)+5$	$i_{b^{(c)}_{rtl}+o}\ \mathbf{U}^{(c)}_0(o)\ \boldsymbol{\beta}^{(c)}(o')\ \mathbb{1}_{s(\mathbf{i})<S}$	$(\mathbf{e}_{b^{(c)}_{srv}+o'} - \mathbf{e}_{b^{(c)}_{rtl}+o})^T$
$7(c-1)+6$	$i_{b^{(c)}_{srv}+o}\ T^{(c)}(o, o')\ \mathbb{1}_{o \neq o'}$	$(\mathbf{e}_{b^{(c)}_{srv}+o'} - \mathbf{e}_{b^{(c)}_{srv}+o})^T$
$7(c-1)+7$	$i_{b^{(c)}_{srv}+o}\ \mathbf{T}^{(c)}_0(o)$	$-\mathbf{e}^T_{b^{(c)}_{srv}+o}$

retrial with probability $\boldsymbol{\xi}^{(c)}(o')$ in phase o' since all servers are busy. The third one corresponds to a synchronizing transition involving submodels $b_{arv}+c-1$ and $b_{srv}^{(c)}+o'$ with an arrival in phase $i_{b_{arv}+c-1}$ and making a transition to phase o at rate $B_1^{(c)}(i_{b_{arv}+c-1}, o)$, joining the pool of servers and starting the PH service with probability $\boldsymbol{\beta}^{(c)}(o')$ in phase o' since there is an idle server. Note that the particular dimension associated with MAP arrivals experiences a phase displacement of $o - i_{b_{arv}+c-1}$ in the first three transitions, and this manifests itself in state change vector $\mathbf{v}^{(\mathbf{k})}$.

The fourth and fifth transitions in Table 4.9 are due to acyclic PH retrials. The fourth one represents a local transition in submodel $b_{rtl}^{(c)}+o$ with a phase change from o to o' at rate $U^{(c)}(o,o')$ and the fifth one represents a synchronizing transition involving submodels $b_{rtl}^{(c)}+o$ and $b_{srv}^{(c)}+o'$ with a successful retrial in phase o at rate $\mathbf{U}_0^{(c)}(o)$, joining the pool of servers and starting the PH service with probability $\boldsymbol{\beta}^{(c)}(o')$ in phase o' since there is an idle server. The last two transitions are due to PH services. The sixth transition class corresponds to a local transition in submodel $b_{srv}^{(c)}+o$ with a phase change from o to o' at rate $T^{(c)}(o,o')$ and the seventh one corresponds to a local transition in submodel $b_{srv}^{(c)}+o$ with a service completion in phase o at rate $\mathbf{T}_0^{(c)}(o)$. Note that in the fourth through seventh transitions, each transition rate $\alpha_k(\mathbf{i})$ incorporates the number of customers present along the particular dimension as a multiplier, which is $i_{b_{rtl}^{(c)}+o}$ in acyclic PH retrials and $i_{b_{srv}^{(c)}+o}$ in PH services.

In the following, we show that the condition

$$\sum_{c=1}^{C} \frac{\lambda^{(c)}}{\mu^{(c)}} < S \tag{4.7}$$

is necessary and sufficient for the existence of steady-state in the C-class MAP/PH/S queueing model with acyclic PH retrials [112]. The result is obtained from criteria based on drifts by choosing suitable Lyapunov functions. The inequality in (4.7) is intuitive since it implies that the sum of the traffic loads $\lambda^{(c)}/\mu^{(c)}$ of customers across C classes should be less than the number of servers, S, for the system to be ergodic and behave stably. Observe that this is equivalent to requiring the *traffic intensity*

$$\rho = \frac{1}{S}\sum_{c=1}^{C} \frac{\lambda^{(c)}}{\mu^{(c)}}$$

to be less than 1. The Lyapunov function used in the sufficiency proof will again enable us to truncate the countably infinite reachable state space $\mathcal{R}$ so that it includes a given steady-state probability mass. But, different than the models considered in the previous two subsections, this time we will also have a necessity proof.

When arrival, service, or retrial processes have phases, Lyapunov functions need to be chosen with care so that phase information is accounted for. Otherwise, a necessary and sufficient stability condition may not be found. In order to arrive at condition (4.7), Lyapunov functions that work for simple models will be extended with additional terms to cope with the complexities of the particular model. To that end, we first introduce two vectors, $\mathbf{u}^{(c)} \in \mathbb{R}^{m_{arv}^{(c)} \times 1}$ and $\mathbf{w}^{(c)} \in \mathbb{R}_{\geqslant 0}^{m_{srv}^{(c)} \times 1}$, whose entries will be used in the additional terms (see Lemmas 1 and 2 in [112]) of the Lyapunov functions.

There exists a unique vector $\mathbf{u}^{(c)} \in \mathbb{R}^{m_{arv}^{(c)} \times 1}$ for MAP with representation $(B_0^{(c)}, B_1^{(c)})$ and $\lambda^{(c)} = \boldsymbol{\theta}^{(c)} B_1^{(c)} \mathbf{e}$ such that

$$(B_0^{(c)} + B_1^{(c)})\mathbf{u}^{(c)} = \lambda^{(c)}\mathbf{e} - B_1^{(c)}\mathbf{e} \quad \text{and} \quad \mathbf{u}^{(c)}\mathbf{e} = 1.$$

The proof follows by showing that a reduced linear system of equations whose coefficient matrix is obtained from the irreducible generator matrix $B_0^{(c)} + B_1^{(c)}$ is consistent [112]. Entries of $\mathbf{u}^{(c)}$ will be used in additional terms to obtain the stability condition based on average arrival rate $\lambda^{(c)}$ instead of phase-dependent arrival rates in $B_1^{(c)}$.

There exists a unique vector $\mathbf{w}^{(c)} \in \mathbb{R}_{\geqslant 0}^{m_{srv}^{(c)} \times 1}$ for PH service distribution with representation $(\boldsymbol{\beta}^{(c)}, T^{(c)})$ and $\mu^{(c)} = (\boldsymbol{\beta}^{(c)}(-T^{(c)})^{-1}\mathbf{e})^{-1}$ such that

$$\mathbf{w}^{(c)} = \mu^{(c)}(-T^{(c)})^{-1}\mathbf{e} \quad \text{and} \quad \boldsymbol{\beta}^{(c)}\mathbf{w}^{(c)} = 1.$$

The existence and nonnegativity of $\mathbf{w}^{(c)}$ follow from the fact that $-T^{(c)}$ is a nonsingular M-matrix [242]. The normalization condition is a result of the definition of $\mu^{(c)}$ [112]. Entries of $\mathbf{w}^{(c)}$ will be used in additional terms to obtain the stability condition based on average service rate $\mu^{(c)}$ instead of phase-dependent service rates in $\mathbf{T}_0^{(c)} = -T^{(c)}\mathbf{e}$.

Example 6. (cntd.) In our model, traffic intensity $\rho = 0.65625$ and vectors $\mathbf{u}^{(c)}$ and $\mathbf{w}^{(c)}$ for $c = 1, 2$ are computed as

$$\mathbf{u}^{(1)} = \begin{pmatrix} 0.25 \\ 0.75 \end{pmatrix}, \quad \mathbf{u}^{(2)} = (1), \quad \mathbf{w}^{(1)} = \begin{pmatrix} 1.4 \\ 0.6 \end{pmatrix}, \quad \mathbf{w}^{(2)} = (1).$$

Now, we are in a position to state the proof of necessity for condition (4.7) which is based on showing that the model is non-ergodic when the condition does not hold [112]. To that end, we employ the next result which appears in [13, 132] for DTMCs. This result is used before to prove a non-ergodicity condition for an S-server retrial queue with exponentially distributed interarrival, service, and retrial times in [131] where a Lyapunov function $g(\mathbf{i})$ linear in the countably infinite variables is utilized.

A CTMC with generator matrix Q is non-ergodic if there exists two constants $\tau, \sigma \in \mathbb{R}$ and a Lyapunov function $g : \mathcal{R} \to \mathbb{R}$ such that

(i) $\sum_{\mathbf{j}\in\mathcal{R}} P(\mathbf{i},\mathbf{j})|g(\mathbf{j}) - g(\mathbf{i})| \leqslant \tau$ for $\mathbf{i} \in \mathcal{R}$ and
(ii) $\sum_{\mathbf{j}\in\mathcal{R}} P(\mathbf{i},\mathbf{j})\,(g(\mathbf{j}) - g(\mathbf{i})) \geqslant 0$ for $\mathbf{i} \in \mathcal{D}$,

where the sets $\mathcal{D} = \{\mathbf{i} \in \mathcal{R} \mid g(\mathbf{i}) > \sigma\}$ and $\mathcal{R}\backslash\mathcal{D}$ are nonempty, and the transition probability matrix $P \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}|\times|\mathcal{R}|}$ associated with the *embedded* DTMC corresponding to the CTMC is given entrywise as

$$P(\mathbf{i},\mathbf{j}) = \begin{cases} Q(\mathbf{i},\mathbf{j})/|Q(\mathbf{i},\mathbf{i})| & \text{if } \mathbf{j} \neq \mathbf{i} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } \mathbf{i},\mathbf{j} \in \mathcal{R}.$$

We also consider a linear Lyapunov function in the countably infinite variables as our starting point, but add constant terms including entries of vectors $\mathbf{u}^{(c)}$ and $\mathbf{w}^{(c)}$ as in

$$g_{ncs}(\mathbf{i}) = \sum_{c=1}^{C} \frac{1}{\mu^{(c)}} \left(\sum_{o=0}^{m_{rtl}^{(c)}-1} i_{b_{rtl}^{(c)}+o} \right) + \sum_{c=1}^{C} \frac{\mathbf{u}^{(c)}(i_{b_{arv}+c-1})}{\mu^{(c)}} + \sum_{c=1}^{C} \frac{1}{\mu^{(c)}} \left(\sum_{o=0}^{m_{srv}^{(c)}-1} \mathbf{w}^{(c)}(o)\; i_{b_{srv}^{(c)}+o} \right).$$

The first term of $g_{ncs}(\mathbf{i})$ is linear, and its second and third terms are added to obtain a phase-independent condition. Note that each of the three terms is a summation in the form of C other terms, each corresponding to a different customer class.

That $g_{ncs}(\mathbf{i})$ satisfies condition (i) above follows from the fact that P has a finite number of nonzero values in row $\mathbf{i}$, each nonzero $P(\mathbf{i},\mathbf{j})$ is upper bounded by 1, and $|g_{ncs}(\mathbf{j}) - g_{ncs}(\mathbf{i})|$ is finite, thereby ensuring that $\sum_{\mathbf{j}\in\mathcal{R}} P(\mathbf{i},\mathbf{j})|g_{ncs}(\mathbf{j}) - g_{ncs}(\mathbf{i})|$ is finite for $\mathbf{i} \in \mathcal{R}$. The proof that condition (ii) above is satisfied by $\sum_{c=1}^{C} \lambda^{(c)}/\mu^{(c)} \geqslant S$, implying non-ergodicity, in other words, necessity of (4.7) for ergodicity, follows from substituting $Q(\mathbf{i},\mathbf{j})/|Q(\mathbf{i},\mathbf{i})|$ in place of $P(\mathbf{i},\mathbf{j})$ in the left-hand side of the inequality in condition (ii) and obtaining

$$\frac{1}{|Q(\mathbf{i},\mathbf{i})|} \sum_{\mathbf{j}\in\mathcal{R}} Q(\mathbf{i},\mathbf{j})(g_{ncs}(\mathbf{j}) - g_{ncs}(\mathbf{i})) = \frac{1}{|Q(\mathbf{i},\mathbf{i})|} \left(\sum_{c=1}^{C} \frac{\lambda^{(c)}}{\mu^{(c)}} - s(\mathbf{i}) \right)$$

after a lengthy sequence of algebraic operations. The particular value

$$\sigma = \max_{\mathbf{i}\in\mathcal{R}} \left(\sum_{c=1}^{C} \frac{\mathbf{u}^{(c)}(i_{b_{arv}+c-1})}{\mu^{(c)}} \right)$$

ensures the nonemptiness of $\mathcal{D}$ and $\mathcal{R}\backslash\mathcal{D}$.

Having discussed the necessity of (4.7) for ergodicity through Lyapunov function $g_{ncs}(\mathbf{i})$, we can move to the construction of Lyapunov function $g_{sfc}(\mathbf{i})$ for sufficiency of the same condition using the result in [314] introduced at the beginning of this section.

For the sufficiency proof of condition (4.7) for ergodicity, we choose a quadratic Lyapunov function $g(\mathbf{i})$ similar to those in [19] to obtain a drift function $d(\mathbf{i})$ with infinite variables having negative coefficients for $\mathbf{i} \in \mathcal{R}$. If customers in the orbits attempt to receive service in all PH retrial phases, adding terms with variables each corresponding to the number of customers in a service phase yields the condition in (4.7). However, if the PH retrial process of a class c customer includes a phase, say o, in which no attempt is made to receive service, that is, $\mathbf{U}_0^{(c)}(o) = 0$, then the coefficient of the corresponding variable, $i_{b_{rtl}^{(c)}+o}$, is positive in $d(\mathbf{i})$ at states with no busy servers. Therefore, the value of $g(\mathbf{i})$ should also depend on the number of customers in retrial phases. This can be realized by adding terms chosen with care so that $d(\mathbf{i})$ at states with no busy servers become negative for a sufficiently large number of customers in the orbits. With this understanding, we introduce vector $\mathbf{w}^{(c)}$ whose entries will be used in the additional term of the Lyapunov function due to PH retrials of class c customers (see Lemma 3 in [112]).

For acyclic PH retrial distribution with representation $(\boldsymbol{\xi}^{(c)}, U^{(c)})$, let

$$\hat{U}^{(c)} = U^{(c)} + \text{diag}(\mathbf{U}_0^{(c)})$$

and $\boldsymbol{\eta}^{(c)} \in \mathbb{R}^{m_{rtl}^{(c)} \times 1}$ be given entrywise as

$$\boldsymbol{\eta}^{(c)}(o) = \begin{cases} -S/\mu^{(c)} & \text{if } o \in \mathcal{I}^{(c)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } o = 0, \ldots, m_{rtl}^{(c)} - 1,$$

where

$$\mathcal{I}^{(c)} = \{o \in \{0, \ldots, m_{rtl}^{(c)} - 1\} \mid \mathbf{U}_0^{(c)}(o) = 0\}.$$

Then there exists $\mathbf{y}^{(c)} \in \mathbb{R}^{m_{rtl}^{(c)} \times 1}$ such that

$$\hat{U}^{(c)} \mathbf{y}^{(c)} = \boldsymbol{\eta}^{(c)}.$$

Note that all entries in row o of $\hat{U}^{(c)}$ are zero if and only if $\hat{U}^{(c)}(o, o) = 0$.

The proof of this result appears in [112] and follows from the row echelon form of upper-triangular $\hat{U}^{(c)}$ obtained by interchanging all zero rows with nonzero rows below them [242], its rank, and that $\hat{U}^{(c)}\mathbf{y}^{(c)} = \boldsymbol{\eta}^{(c)}$ is consistent with infinitely many solutions [242]. Note that negative entries of $\boldsymbol{\eta}^{(c)}$ could take a smaller value than $-S/\mu^{(c)}$ that may lead to a smaller truncated state space, but this value is chosen in order to bound coefficients of infinite variables from above by $\sum_{c=1}^{C} \lambda^{(c)}/\mu^{(c)} - S$ in $d(\mathbf{i})$ for all states. Since there are infinitely many solutions to $\hat{U}^{(c)}\mathbf{y}^{(c)} = \boldsymbol{\eta}^{(c)}$, we choose to set $\mathbf{y}^{(c)}(o)$ to 1 if row o of $\hat{U}^{(c)}$ is zero. The entries of vector $\boldsymbol{\eta}^{(c)}$ contribute to coefficients of infinite

variables associated with retrials of class c customers in $d(\mathbf{i})$. Therefore, its entries need to be nonpositive with at least one negative entry when $\mathcal{I}^{(c)} \neq \varnothing$. We remark that the existence of at least one negative entry in $\boldsymbol{\eta}^{(c)}$ requires $\mathbf{U}_0^{(c)}$ to have at least one zero entry.

Example 6. (cont'd) In our model,

$$\mathcal{I}^{(1)} = \{0\}, \quad \boldsymbol{\eta}^{(1)} = \begin{pmatrix} -1.25 \\ 0 \end{pmatrix} \quad \text{and} \quad \mathcal{I}^{(2)} = \varnothing, \quad \boldsymbol{\eta}^{(2)} = (0)\,.$$

Hence,

$$\mathbf{y}^{(1)} = \begin{pmatrix} 2.25 \\ 1 \end{pmatrix} \quad \text{and} \quad \mathbf{y}^{(2)} = (1)\,.$$

If PH retrials are allowed to be non-acyclic, $\hat{U}^{(c)}\mathbf{y}^{(c)} = \boldsymbol{\eta}^{(c)}$ may very well be inconsistent since $\hat{U}^{(c)}$ is a singular M-matrix [30]. For instance, consider

$$U^{(c)} = \begin{pmatrix} -2 & 1 \\ 2 & -2 \end{pmatrix} \quad \text{for which} \quad \hat{U}^{(c)} = \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\eta}^{(c)} = \begin{pmatrix} 0 \\ -S/\mu^{(c)} \end{pmatrix}.$$

In this case, vector $\mathbf{y}^{(c)}$ does not exist since the linear system is inconsistent.

The particular Lyapunov function used [112] is given by

$$\begin{aligned} g_{sfc}(\mathbf{i}) = &\sum_{c=1}^{C}\sum_{c'=1}^{C} \frac{1}{2\mu^{(c)}\mu^{(c')}} \left(\sum_{o=0}^{m_{rtl}^{(c)}-1} i_{b_{rtl}^{(c)}+o}\right)\left(\sum_{o'=0}^{m_{rtl}^{(c')}-1} i_{b_{rtl}^{(c')}+o'}\right) \\ &+ \sum_{c=1}^{C}\left(\sum_{c'=1}^{C} \frac{\mathbf{u}^{(c')}(i_{b_{arv}+c'-1})}{\mu^{(c)}\mu^{(c')}}\right)\left(\sum_{o=0}^{m_{rtl}^{(c)}-1} i_{b_{rtl}^{(c)}+o}\right) \\ &+ \sum_{c=1}^{C}\left(\sum_{c'=1}^{C} \frac{1}{\mu^{(c)}\mu^{(c')}}\left(\sum_{o=0}^{m_{srv}^{(c')}-1} \mathbf{w}^{(c')}(o)\; i_{b_{srv}^{(c')}+o}\right)\right)\left(\sum_{o'=0}^{m_{rtl}^{(c)}-1} i_{b_{rtl}^{(c)}+o'}\right) \\ &+ \sum_{c=1}^{C}\sum_{o=0}^{m_{rtl}^{(c)}-1} \mathbf{y}^{(c)}(o)\; i_{b_{rtl}^{(c)}+o} + \sum_{c=1}^{C} \zeta^{(c)}(\mathbf{i})\left(\sum_{o=0}^{m_{srv}^{(c)}-1} i_{b_{srv}^{(c)}+o}\right), \end{aligned}$$

where for $c = 1, \ldots, C$

$$\begin{aligned} \zeta^{(c)}(\mathbf{i}) = &\min_{o \notin \mathcal{I}^{(c)}} \left(z^{(c)}(o,\mathbf{i}) - S/(\mathbf{U}_0^{(c)}(o)\mu^{(c)})\right) \quad \text{and} \\ z^{(c)}(o,\mathbf{i}) = &\; \mathbf{y}^{(c)}(o) + \frac{1}{2\mu^{(c)}\mu^{(c)}} + \sum_{c'=1}^{C} \frac{\mathbf{u}^{(c')}(i_{b_{arv}+c'-1})}{\mu^{(c)}\mu^{(c')}} \\ &+ \sum_{c'=1}^{C} \frac{1}{\mu^{(c)}\mu^{(c')}}\left(\sum_{o'=0}^{m_{srv}^{(c')}-1} \mathbf{w}^{(c')}(o')\; i_{b_{srv}^{(c')}+o'}\right) \quad \text{for } o \notin \mathcal{I}^{(c)}. \end{aligned}$$

The first term of $g_{sfc}(\mathbf{i})$ is quadratic, and its second and third terms are added to obtain a phase-independent condition. The last two terms of $g_{sfc}(\mathbf{i})$ are added due to PH retrials. Note that, $\zeta^{(c)}(\mathbf{i})$ is well defined since $\mathcal{I}^{(c)} \neq \{0, \ldots, m_{rtl}^{(c)} - 1\}$ for $c = 1, \ldots, C$. Then $g_{sfc}(\mathbf{i})$ is a quadratic polynomial in which the coefficients of all infinite variables are finite and the coefficient of each quadratic term $i_{b_{rtl}^{(c)}+o} i_{b_{rtl}^{(c')}+o'}$ is positive for $o = 0, \ldots, m_{rtl}^{(c)} - 1$, $o' = 0, \ldots, m_{rtl}^{(c')} - 1$, and $c, c' = 1, \ldots, C$. Hence, $g_{sfc}(\mathbf{i})$ is lower bounded, implying a finite set $\{\mathbf{i} \in \mathcal{R} \mid g_{sfc}(\mathbf{i}) \leqslant r\}$ for all $r < \infty$ in condition (iii) at the beginning of this section. Now, we are in a position to provide the drift from (4.3).

For $s(\mathbf{i}) = S$, the drift is [112]

$$
\begin{aligned}
d(\mathbf{i}) = & \sum_{c=1}^{C} \frac{1}{\mu^{(c)}} \left(\sum_{c'=1}^{C} \frac{\lambda^{(c')}}{\mu^{(c')}} - S \right) \left(\sum_{o=0}^{m_{rtl}^{(c)}-1} i_{b_{rtl}^{(c)}+o} \right) - \sum_{c=1}^{C} \frac{S}{\mu^{(c)}} \left(\sum_{o \in \mathcal{I}^{(c)}} i_{b_{rtl}^{(c)}+o} \right) \\
& + \sum_{c=1}^{C} \sum_{o=0}^{m_{arv}^{(c)}-1} B_1^{(c)}(i_{b_{arv}+c-1}, o) \left[\frac{\mathbf{u}^{(c)}(o)}{\mu^{(c)}\mu^{(c)}} + \sum_{\substack{c'=1 \\ c' \neq c}}^{C} \frac{\mathbf{u}^{(c')}(i_{b_{arv}+c'-1})}{\mu^{(c)}\mu^{(c')}} \right] \\
& + \sum_{c=1}^{C} \sum_{o=0}^{m_{arv}^{(c)}-1} B_1^{(c)}(i_{b_{arv}+c-1}, o) \left[\sum_{c'=1}^{C} \frac{1}{\mu^{(c)}\mu^{(c')}} \left(\sum_{o'=0}^{m_{srv}^{(c')}-1} \mathbf{w}^{(c')}(o')\, i_{b_{srv}^{(c')}+o'} \right) \right] \\
& + \sum_{c=1}^{C} \sum_{o=0}^{m_{arv}^{(c)}-1} B_1^{(c)}(i_{b_{arv}+c-1}, o) \left[\left(\sum_{o'=0}^{m_{rtl}^{(c)}-1} \boldsymbol{\xi}^{(c)}(o') \mathbf{y}^{(c)}(o') \right) + \frac{1}{2\mu^{(c)}\mu^{(c)}} \right] \\
& - \sum_{c=1}^{C} \zeta^{(c)}(\mathbf{i}) \left(\sum_{o=0}^{m_{srv}^{(c)}-1} i_{b_{srv}^{(c)}+o} \mathbf{T}_0^{(c)}(o) \right);
\end{aligned}
$$

and for $s(\mathbf{i}) < S$, the drift is [112]

$$
\begin{aligned}
d(\mathbf{i}) = & \sum_{c=1}^{C} \frac{1}{\mu^{(c)}} \left(\sum_{c'=1}^{C} \frac{\lambda^{(c')}}{\mu^{(c')}} - \sum_{c'=1}^{C} \sum_{o=0}^{m_{srv}^{(c')}-1} i_{b_{srv}^{(c')}+o} \right) \left(\sum_{o'=0}^{m_{rtl}^{(c)}-1} i_{b_{rtl}^{(c)}+o'} \right) \\
& - \sum_{c=1}^{C} \frac{S}{\mu^{(c)}} \left(\sum_{o \in \mathcal{I}^{(c)}} i_{b_{rtl}^{(c)}+o} \right) \\
& + \sum_{c=1}^{C} \sum_{o \notin \mathcal{I}^{(c)}} i_{b_{rtl}^{(c)}+o}\, \mathbf{U}_0^{(c)}(o) \left(-z^{(c)}(o, \mathbf{i}) + \zeta^{(c)}(\mathbf{i}) \right) \\
& + \sum_{c=1}^{C} \zeta^{(c)}(\mathbf{i}) \left(\sum_{o=0}^{m_{arv}^{(c)}-1} B_1^{(c)}(i_{b_{arv}+c-1}, o) - \sum_{o'=0}^{m_{srv}^{(c)}-1} i_{b_{srv}^{(c)}+o'}\, \mathbf{T}_0^{(c)}(o') \right).
\end{aligned}
$$

When $\sum_{c=1}^{C}(\lambda^{(c)}/\mu^{(c)}) < S$, coefficients of all infinite variables are negative; hence, $d(\mathbf{i}) < \infty$ for all $\mathbf{i} \in \mathcal{C} \subset \mathcal{R}$ and condition (ii) at the beginning of this section holds. Besides, $\mathcal{C} = \{\mathbf{i} \in \mathcal{R} \mid d(\mathbf{i}) > -\gamma\}$ is finite for any arbitrarily chosen $\gamma > 0$. Therefore, condition (i) at the beginning of this section also holds.

Example 6. (cont'd) In our model,

$$\begin{aligned}
\zeta^{(1)}(\mathbf{i}) &= z^{(1)}(1,\mathbf{i}) - 1.25,\\
z^{(1)}(1,\mathbf{i}) &= 2.4453125 + 0.390625\mathbf{u}^{(1)}(i_4)\\
&\quad + 0.546875i_6 + 0.234375i_7 + 1.25i_8,\\
\zeta^{(2)}(\mathbf{i}) &= z^{(2)}(0,\mathbf{i}) - 8,\\
z^{(2)}(0,\mathbf{i}) &= 7 + 1.25\mathbf{u}^{(1)}(i_4) + 1.75i_6 + 0.75i_7 + 4i_8.
\end{aligned}$$

The Lyapunov function is given by

$$\begin{aligned}
g(\mathbf{i}) &= 0.1953125(i_1+i_2)^2 + 1.25(i_1+i_2)i_3 + 2i_3^2 + 2.25i_1 + i_2 + i_3\\
&\quad + \left(0.390625\mathbf{u}^{(1)}(i_4) + 1.25\mathbf{u}^{(2)}(i_5)\right)(i_1+i_2)\\
&\quad + (0.546875i_6 + 0.234375i_7 + 1.25i_8)(i_1+i_2)\\
&\quad + \left(1.25\mathbf{u}^{(1)}(i_4) + 4\mathbf{u}^{(2)}(i_5) + 1.75i_6 + 0.75i_7 + 4i_8\right)i_3\\
&\quad + \zeta^{(1)}(\mathbf{i})(i_6+i_7) + \zeta^{(2)}(\mathbf{i})i_8.
\end{aligned}$$

The drift for $s(\mathbf{i}) = 2$ is

$$\begin{aligned}
d(\mathbf{i}) &= -1.6796875i_1 - 0.4296875i_2 - 1.375i_3\\
&\quad + \mathbb{1}_{i_4=1}(0.546875i_6 + 0.234375i_7 + 1.25i_8 + 3.79296875)\\
&\quad + 0.5\mathbf{u}^{(1)}(i_4) + 0.7i_6 + 0.3i_7 + 1.6i_8 + 2.8\\
&\quad + \zeta^{(1)}(\mathbf{i})(-i_6 - 3i_7) + \zeta^{(2)}(\mathbf{i})(-0.5i_8),
\end{aligned}$$

and the drift for $s(\mathbf{i}) < 2$ is

$$\begin{aligned}
d(\boldsymbol{x}) &= (-0.6875 - i_6 - i_7 - i_8)(0.625i_1 + 0.625i_2 + 2i_3)\\
&\quad + \zeta^{(1)}(\mathbf{i})(\mathbb{1}_{i_4=1} - i_6 - 3i_7) + \zeta^{(2)}(\mathbf{i})(0.4 - 0.5i_8).
\end{aligned}$$

The global maximum drift is $\chi = 4.9359375$ and attained when $i_6 + i_8 = 2$ at state $(0,0,0,1,0,1,0,1)$. With a lower bound of 90% on the steady-state probability mass (i.e., $\varepsilon = 0.1$), we compute $\gamma = 44.4234$ and $|\mathcal{C}| = 189{,}558$.

Now, we provide the truncated multidimensional CTMC and a Kronecker representation of its generator matrix. A block tridiagonal generator matrix as in the previous two models of this section will not be used since the order of diagonal blocks increases considerably with a large number of dimensions

as the partition index increases, and such a block partitioning does not yield itself to a scalable steady-state solver.

Traditionally, truncation of a countably infinite reachable state space for exponentially distributed retrial times is achieved by imposing an upper bound, say M, on the total number of customers in the orbit [131, 301]. This implies blocking of an arriving customer if the total number of retrial customers is already M. We consider a similar truncated model in which the size of retrial phase o of orbit c is limited by

$$N_o^{(c)} = \max\{i_{b_{rtl}^{(c)}+o} \mid \mathbf{i} \in \mathcal{C}\} + 1 \quad \text{for } o = 0, \ldots, m_{rtl}^{(c)} - 1 \text{ and } c = 1, \ldots, C,$$

and the truncated reachable state space is given by

$$\bar{\mathcal{R}} = \bar{\mathcal{S}}_{rtl} \times \mathcal{S}_{arv} \times \mathcal{R}_{srv} \quad \text{with} \quad \bar{\mathcal{S}}_{rtl} = \bigtimes_{c=1}^{C} \left(\bigtimes_{o=0}^{m_{rtl}^{(c)}-1} \{0, \ldots, N_o^{(c)} - 1\} \right).$$

We consider the multidimensional CTMC with finite state space $\bar{\mathcal{R}}$ and truncated generator matrix $\bar{Q}$ given entrywise as

$$\bar{Q}(\mathbf{i}, \mathbf{j}) = \begin{cases} Q(\mathbf{i}, \mathbf{j}) & \text{if } \mathbf{i} \neq \mathbf{j} \\ -\sum_{\mathbf{j}' \neq \mathbf{i}} \bar{Q}(\mathbf{i}, \mathbf{j}') & \text{otherwise} \end{cases} \quad \text{for} \quad \mathbf{i}, \mathbf{j} \in \bar{\mathcal{R}}.$$

The truncated generator matrix $\bar{Q}$ cannot be represented as sums of Kronecker products since $\bar{\mathcal{R}}$ is not a Cartesian product of subsets of submodel state spaces [43]. Therefore, we first let $N = |\mathcal{R}_{srv}|$ and define the function $\varphi : \mathcal{R}_{srv} \to \{0, \ldots, N-1\}$ that determines the lexicographical order of states in $\mathcal{R}_{srv}$ as

$$\varphi(i_{b_{srv}^{(1)}}, \ldots, i_H) = i_H + \sum_{c=1}^{m_{srv}-1} \sum_{o=0}^{i_{H_I+m_{arv}+c}-1} \sum_{s=0}^{S-r^{(c)}-o} \frac{(s + m_{srv} - 1 - c)!}{s!\ (m_{srv} - 1 - c)!},$$

where $r^{(c)} = \sum_{c'=1}^{c-1} i_{b_{srv}^{(1)}+c'-1}$ for $(i_{b_{srv}^{(1)}}, \ldots, i_H) \in \mathcal{R}_{srv}$. This function is one-to-one and onto. Hence, its inverse $\varphi^{-1} : \{0, \ldots, N-1\} \to \mathcal{R}_{srv}$ is well defined. Second, we let

$$\bar{\mathcal{R}}_p = \bar{\mathcal{S}}_{rtl} \times \mathcal{S}_{arv} \times \{\varphi^{-1}(p)\} \quad \text{for } p = 0, \ldots, N-1.$$

Then $\bar{\mathcal{R}}_0, \ldots, \bar{\mathcal{R}}_{N-1}$ is a partitioning of $\bar{\mathcal{R}}$ such that

$$\bigcup_{p=0}^{N-1} \bar{\mathcal{R}}_p = \bar{\mathcal{R}} \quad \text{and} \quad \bar{\mathcal{R}}_p \cap \bar{\mathcal{R}}_w = \varnothing \quad \text{for } p \neq w \text{ and } p, w = 0, \ldots, N-1.$$

Finally, we have the block partitioning of the truncated generator matrix in

$$\bar{Q} = \begin{pmatrix} \bar{Q}(0,0) & \dots & \bar{Q}(0,N-1) \\ \vdots & \ddots & \vdots \\ \bar{Q}(N-1,0) & \dots & \bar{Q}(N-1,N-1) \end{pmatrix}$$

where block $\bar{Q}(p,w)$ includes transition rates from states in $\bar{\mathcal{R}}_p$ to states in $\bar{\mathcal{R}}_w$.

Example 6. (cont'd) In our model, the particular set $\mathcal{C}$ yields

$$N_0^{(1)} = 83, \quad N_1^{(1)} = 89, \quad \text{and} \quad N_0^{(2)} = 28.$$

Since we already have $|\mathcal{S}_{arv}| = 2$ and $|\mathcal{R}_{srv}| = 10$, the size of the truncated reachable state space becomes

$$|\bar{\mathcal{R}}| = N_0^{(1)} \times N_1^{(1)} \times N_0^{(2)} \times |\mathcal{S}_{arv}| \times |\mathcal{R}_{srv}| = 4{,}136{,}720.$$

Furthermore, $N = 10$ and $\varphi(0,0,0) = 0$, $\varphi(0,0,1) = 1$, $\varphi(0,0,2) = 2$, $\varphi(0,1,0) = 3$, $\varphi(0,1,1) = 4$, $\varphi(0,2,0) = 5$, $\varphi(1,0,0) = 6$, $\varphi(1,0,1) = 7$, $\varphi(1,1,0) = 8$, and $\varphi(2,0,0) = 9$. Hence, $\bar{Q}$ is a (10×10) block matrix consisting of $N^2 = 100$ blocks. Note that many states which do not appear in $\mathcal{C}$ are included in $\bar{\mathcal{R}}$ with this truncation approach at the expense of blocks of the same order.

Now, we are in a position to represent blocks $\bar{Q}(p,w)$ of truncated generator matrix $\bar{Q}$ as sums of Kronecker products of smaller matrices. Since retrial phases are distributed to different dimensions and there are multiple customer classes in this model, it is more convenient to represent the blocks as sums of Kronecker products of auxiliary matrices which will also be given as Kronecker products of the smaller transition matrices.

To that end, we first define auxiliary matrices $W_{\tilde{B}_0^{(c)}}, W_{B_1^{(c)}} \in \mathbb{R}_{\geqslant 0}^{|\mathcal{S}_{arv}| \times |\mathcal{S}_{arv}|}$ with transition rates among submodels corresponding to phases of MAPs for $c = 1, \dots, C$ as

$$W_{\tilde{B}_0^{(c)}} = \left(\bigotimes_{c'=1}^{c-1} I_{m_{arv}^{(c')}} \right) \otimes \tilde{B}_0^{(c)} \otimes \left(\bigotimes_{c'=c+1}^{C} I_{m_{arv}^{(c')}} \right),$$

$$W_{B_1^{(c)}} = \left(\bigotimes_{c'=1}^{c-1} I_{m_{arv}^{(c')}} \right) \otimes B_1^{(c)} \otimes \left(\bigotimes_{c'=c+1}^{C} I_{m_{arv}^{(c')}} \right),$$

where matrix $\tilde{B}_0^{(c)} \in \mathbb{R}_{\geqslant 0}^{m_{arv}^{(c)} \times m_{arv}^{(c)}}$ consists of the off-diagonal entries of $B_0^{(c)}$. Note that $W_{\tilde{B}_0^{(c)}}$ and $W_{B_1^{(c)}}$ are formed of $m_{arv} = C$ Kronecker products and are associated with submodels $H_I + 1$ through $H_I + m_{arv}$. In particular, $W_{\tilde{B}_0^{(c)}}$ represents phase changes without arrival in MAP of class c customers and is related to transition $7(c-1)+1$ in Table 4.9, whereas $W_{B_1^{(c)}}$ represents phase changes with arrival in MAP of class c customers and is part of transitions

$7(c-1)+2$ and $7(c-1)+3$ depending on whether all servers are busy or not. The function $\mathbb{1}_{o \neq i_{b_{arv}+c-1}}$ in transition $7(c-1)+1$ is handled by using $\tilde{B}_0^{(c)}$ instead of $B_0^{(c)}$ in the definition of $W_{\tilde{B}_0^{(c)}}$.

Second, we introduce

$$E_l^- = \text{subdiag}((1,\ldots,l-1)^T)_{l\times l} \quad \text{and} \quad E_l^+ = \text{supdiag}(\mathbf{e})_{l\times l} \quad \text{for } l \in \mathbb{Z}_{>0}.$$

associated with acyclic PH retrial processes so that $E_l^- \in \mathbb{R}_{\geqslant 0}^{l\times l}$ stores the multipliers of phase changes and successful completions, while $E_l^+ \in \mathbb{R}_{\geqslant 0}^{l\times l}$ stores the multipliers of starts. These two matrices represent, respectively, a decrease and an increase by one in the number of customers of the particular retrial phase, with the rate of decrease being multiplied by the number of customers in that phase.

We define auxiliary matrices $A_o^{(c)}, R_{o,o'}^{(c)}, Z_o^{(c)} \in \mathbb{R}_{\geqslant 0}^{|\bar{\mathcal{S}}_{rtl}|\times|\bar{\mathcal{S}}_{rtl}|}$ with transition rates among submodels corresponding to phases of acyclic PH retrial processes using E_l^- and E_l^+ as

$$A_o^{(c)} = \bigotimes_{c'=1}^{C} \bigotimes_{o'=0}^{m_{rtl}^{(c')}-1} A_{o,o'}^{(c,c')}, \quad R_{o,o'}^{(c)} = \bigotimes_{c'=1}^{C} \bigotimes_{o''=0}^{m_{rtl}^{(c')}-1} R_{o,o',o''}^{(c,c')}, \quad \text{and}$$

$$Z_o^{(c)} = \bigotimes_{c'=1}^{C} \bigotimes_{o'=0}^{m_{rtl}^{(c')}-1} Z_{o,o'}^{(c,c')},$$

where

$$A_{o,o'}^{(c,c')} = \begin{cases} E^+_{N_{o'}^{(c')}} & \text{if } c'=c \text{ and } o'=o \\ I_{N_{o'}^{(c')}} & \text{otherwise} \end{cases}$$

$$R_{o,o',o''}^{(c,c')} = \begin{cases} E^-_{N_{o''}^{(c')}} & \text{if } c'=c \text{ and } o''=o \\ E^+_{N_{o''}^{(c')}} & \text{if } c'=c \text{ and } o''=o' \\ I_{N_{o''}^{(c')}} & \text{otherwise} \end{cases},$$

$$Z_{o,o'}^{(c,c')} = \begin{cases} E^-_{N_{o'}^{(c')}} & \text{if } c'=c \text{ and } o'=o \\ I_{N_{o'}^{(c')}} & \text{otherwise} \end{cases}$$

for $o, o', o'' = 0,\ldots,m_{rtl}^{(c')}-1$, $o \neq o'$, and $c, c' = 1,\ldots,C$. Note that $A_o^{(c)}$, $R_{o,o'}^{(c)}$, and $Z_o^{(c)}$ consist of m_{rtl} Kronecker products and are associated with submodels 1 through H_I. In particular, $A_o^{(c)}$ represents MAP arrivals that join orbit c to start the acyclic PH retrial process in phase o due to having all servers busy and is related to transition $7(c-1)+2$ in Table 4.9. Matrix

$R^{(c)}_{o,o'}$ represents the acyclic PH retrial phase change of a class c customer from o to o' and is related to transition $7(c-1)+4$, and $Z^{(c)}_o$ represents the successful retrial process completion of a class c customer in phase o and is related to transition $7(c-1)+5$.

Now, we are ready to provide a Kronecker representation for the nonzero blocks of $\bar{Q}$. For $p, w = 0, \ldots, N-1$, block (p,w) of $\bar{Q}$ can be expressed as

$$\bar{Q}(p,w) = \begin{cases} \sum_{c=1}^{C} (I_{|\bar{\mathcal{S}}_{rtl}|} \otimes W_{\tilde{B}_0^{(c)}}) \\ +\mathbb{1}_{\bar{s}(p)=S} \sum_{c=1}^{C} \sum_{o'=0}^{m_{rtl}^{(c)}-1} \boldsymbol{\xi}^{(c)}(o')(A^{(c)}_{o'} \otimes W_{B_1^{(c)}}) \\ + \sum_{c=1}^{C} \sum_{o=0}^{m_{rtl}^{(c)}-1} \sum_{\substack{o'=0 \\ o' \neq o}}^{m_{rtl}^{(c)}-1} U^{(c)}(o,o')(R^{(c)}_{o,o'} \otimes I_{|\mathcal{S}_{arv}|}) \\ +\bar{Q}_D(p,p) \\ \qquad\qquad \text{if } \mathbf{v}_{srv}(p,w) = \mathbf{0} \\ \\ \mathbb{1}_{\bar{s}(p)<S}\ \boldsymbol{\beta}^{(c)}(o')(I_{|\bar{\mathcal{S}}_{rtl}|} \otimes W_{B_1^{(c)}}) \\ +\mathbb{1}_{\bar{s}(p)<S} \sum_{o=0}^{m_{rtl}^{(c)}-1} \mathbf{U}_0^{(c)}(o)\boldsymbol{\beta}^{(c)}(o')(Z^{(c)}_o \otimes I_{|\mathcal{S}_{arv}|}) \\ \qquad\qquad \text{if } \mathbf{v}_{srv}(p,w) = \mathbf{e}^T_{\bar{b}^{(c)}_{srv}+o'} \\ \\ i^{(srv)}_{\bar{b}^{(c)}_{srv}+o}(p)\ T^{(c)}(o,o')(I_{|\bar{\mathcal{S}}_{rtl}|} \otimes I_{|\mathcal{S}_{arv}|}) \\ \qquad\qquad \text{if } (\mathbf{v}_{srv}(p,w) = (\mathbf{e}_{\bar{b}^{(c)}_{srv}+o'} - \mathbf{e}_{\bar{b}^{(c)}_{srv}+o})^T \\ \qquad\qquad\quad \text{and } o \neq o') \\ \\ i^{(srv)}_{\bar{b}^{(c)}_{srv}+o}(p)\ \mathbf{T}_0^{(c)}(o)(I_{|\bar{\mathcal{S}}_{rtl}|} \otimes I_{|\mathcal{S}_{arv}|}) \\ \qquad\qquad \text{if } \mathbf{v}_{srv}(p,w) = -\mathbf{e}^T_{\bar{b}^{(c)}_{srv}+o} \\ \\ 0 \\ \qquad\qquad \text{otherwise} \end{cases},$$

where

$$\mathbf{v}_{srv}(p,w) = \varphi^{-1}(w) - \varphi^{-1}(p), \quad \mathbf{i}^{(srv)}(p) = \varphi^{-1}(p), \quad \bar{s}(p) = \mathbf{i}^{(srv)}(p)\mathbf{e},$$

$$\bar{b}^{(c)}_{srv} = 1 + \sum_{c'=1}^{c-1} m^{(c')}_{srv}, \quad |\bar{\mathcal{S}}_{rtl}| = \prod_{c=1}^{C} \prod_{o=0}^{m_{rtl}^{(c)}-1} N_o^{(c)}, \quad \text{and} \quad |\mathcal{S}_{arv}| = \prod_{c=1}^{C} m^{(c)}_{arv}.$$

Here, $\mathbf{v}_{srv}(p,w) \in \mathbb{Z}^{1 \times m_{srv}}$ is the state change vector restricted to the difference between states $\varphi^{-1}(w) \in \mathcal{R}_{srv}$ and $\varphi^{-1}(p) \in \mathcal{R}_{srv}$. Vector $\mathbf{i}^{(srv)}(p) \in \mathbb{Z}^{1 \times m_{srv}}$ is that part of $\mathbf{i} \in \mathbb{Z}^{1 \times H}$ corresponding to the submodels associated with the PH service process in truncated reachable state space partition $\bar{\mathcal{R}}_p$. The integers $\bar{s}(p)$ and $\bar{b}^{(c)}_{srv}$ are, respectively, the number of busy servers and the index in $\mathbf{i}^{(srv)}(p)$ assigned to phase 0 of the PH service process of class c customers.

When $\mathbf{v}_{srv}(p,w)=\mathbf{0}$, it must be that $p=w$; hence, transitions $7(c-1)+1$, $7(c-1)+2$, and $7(c-1)+4$ with $c=1,\ldots,C$ in Table 4.9 contribute to the diagonal blocks $\bar{Q}(p,p)$ for $p=0,\ldots,N-1$. There is no change in the allocation of servers to customers in this case. As before, we leave out the definition of $\bar{Q}_D(p,p)$. On the other hand, when $\mathbf{v}_{srv}(p,w)=\mathbf{e}^T_{\bar{b}^{(c)}_{srv}+o'}$, number of servers associated with phase o' of the PH service process of class c customers increases by one. This can only happen in an off-diagonal block and transitions $7(c-1)+3$ and $7(c-1)+5$ contribute to those off-diagonal blocks. When $\mathbf{v}_{srv}(p,w)=(\mathbf{e}_{\bar{b}^{(c)}_{srv}+o'}-\mathbf{e}_{\bar{b}^{(c)}_{srv}+o})^T$ and $o\neq o'$, number of servers associated with phase o' of the PH service process of class c customers increases by one and number of servers associated with phase o of the PH service process of class c customers decreases by one. Although the total number of servers allocated to class c customers do not change, this can only take place in an off-diagonal block. Transition $7(c-1)+6$ contributes to those off-diagonal blocks. Finally, when $\mathbf{v}_{srv}(p,w)=-\mathbf{e}^T_{\bar{b}^{(c)}_{srv}+o}$, number of servers associated with phase o of the PH service process of class c customers decreases by one. This can only happen in an off-diagonal block and transition $7(c-1)+7$ contributes to those off-diagonal blocks.

We remark that all possible transitions in Table 4.9 are covered with the given Kronecker representation. For the last two transitions, the number of class c customers in the particular phase of the PH service process, $i^{(srv)}_{\bar{b}^{(c)}_{srv}+o}(p)$, appears explicitly as a multiplier in the expressions for $\bar{Q}(p,w)$. Observe that the number of class c customers in the particular phase of the acyclic PH retrial process appears implicitly in the expressions involving $R^{(c)}_{o,o'}$ and $Z^{(c)}_o$ through E^-_l.

Example 6. (cont'd) In our model, the auxiliary matrices are given by

$$\tilde{B}^{(1)}_0=\begin{pmatrix}0&1\\0&0\end{pmatrix},\quad \tilde{B}^{(2)}_0=(0),\quad W_{\tilde{B}^{(1)}_0}=\tilde{B}^{(1)}_0\otimes I_1,\quad W_{\tilde{B}^{(2)}_0}=I_2\otimes\tilde{B}^{(2)}_0,$$

$$W_{B^{(1)}_1}=B^{(1)}_1\otimes I_1,\quad W_{B^{(2)}_1}=I_2\otimes B^{(2)}_1,\quad A^{(1)}_0=E^+_{N^{(1)}_0}\otimes I_{N^{(1)}_1}\otimes I_{N^{(2)}_0},$$

$$A^{(1)}_1=I_{N^{(1)}_0}\otimes E^+_{N^{(1)}_1}\otimes I_{N^{(2)}_0},\quad A^{(2)}_0=I_{N^{(1)}_0}\otimes I_{N^{(1)}_1}\otimes E^+_{N^{(2)}_0},$$

$$R^{(1)}_{0,1}=E^-_{N^{(1)}_0}\otimes E^+_{N^{(1)}_1}\otimes I_{N^{(2)}_0},\quad R^{(1)}_{1,0}=E^+_{N^{(1)}_0}\otimes E^-_{N^{(1)}_1}\otimes I_{N^{(2)}_0},$$

$$Z^{(1)}_0=E^-_{N^{(1)}_0}\otimes I_{N^{(1)}_1}\otimes I_{N^{(2)}_0},\quad Z^{(1)}_1=I_{N^{(1)}_0}\otimes E^-_{N^{(1)}_1}\otimes I_{N^{(2)}_0},$$

$$Z^{(2)}_0=I_{N^{(1)}_0}\otimes I_{N^{(1)}_1}\otimes E^-_{N^{(2)}_0}.$$

In our model, $|\bar{\mathcal{S}}_{rtl}| = 206{,}836$ and $|\mathcal{S}_{arv}| = 2$, and the 62 nonzero blocks of $\bar{Q}$ are given by

$$\begin{aligned}
\bar{Q}(0,0) - \bar{Q}_D(0,0) &= \bar{Q}(1,1) - \bar{Q}_D(1,1) = \bar{Q}(3,3) - \bar{Q}_D(3,3) \\
&= \bar{Q}(6,6) - \bar{Q}_D(6,6) \\
&= I_{|\bar{\mathcal{S}}_{rtl}|} \otimes W_{\tilde{B}_0^{(1)}} + R_{0,1}^{(1)} \otimes I_{|\mathcal{S}_{arv}|},
\end{aligned}$$

$$\begin{aligned}
\bar{Q}(2,2) - \bar{Q}_D(2,2) &= \bar{Q}(4,4) - \bar{Q}_D(4,4) = \bar{Q}(5,5) - \bar{Q}_D(5,5) \\
&= \bar{Q}(7,7) - \bar{Q}_D(7,7) = \bar{Q}(8,8) - \bar{Q}_D(8,8) \\
&= \bar{Q}(9,9) - \bar{Q}_D(9,9) \\
&= I_{|\bar{\mathcal{S}}_{rtl}|} \otimes W_{\tilde{B}_0^{(1)}} + A_0^{(1)} \otimes W_{B_1}^{(1)} + A_0^{(2)} \otimes W_{B_1}^{(2)} \\
&\quad + R_{0,1}^{(1)} \otimes I_{|\mathcal{S}_{arv}|},
\end{aligned}$$

$$\begin{aligned}
\bar{Q}(0,1) &= \bar{Q}(1,2) = \bar{Q}(3,4) = \bar{Q}(6,7) \\
&= I_{|\bar{\mathcal{S}}_{rtl}|} \otimes W_{B_1^{(2)}} + 0.5 Z_0^{(2)} \otimes I_{|\mathcal{S}_{arv}|},
\end{aligned}$$

$$\begin{aligned}
\bar{Q}(0,3) &= \bar{Q}(1,4) = \bar{Q}(3,5) = \bar{Q}(6,8) = \bar{Q}(0,6) = \bar{Q}(1,7) \\
&= \bar{Q}(3,8) = \bar{Q}(6,9) \\
&= 0.5 I_{|\bar{\mathcal{S}}_{rtl}|} \otimes W_{B_1^{(1)}} + 0.5 Z_1^{(1)} \otimes I_{|\mathcal{S}_{arv}|},
\end{aligned}$$

$$\bar{Q}(3,0) = \bar{Q}(4,1) = \bar{Q}(8,6) = 3 I_{|\bar{\mathcal{S}}_{rtl}| \cdot |\mathcal{S}_{arv}|},$$

$$\bar{Q}(6,3) = \bar{Q}(7,4) = \bar{Q}(8,5) = 0.25 I_{|\bar{\mathcal{S}}_{rtl}| \cdot |\mathcal{S}_{arv}|},$$

$$\bar{Q}(6,0) = \bar{Q}(7,1) = \bar{Q}(8,3) = \bar{Q}(2,1) = I_{|\bar{\mathcal{S}}_{rtl}| \cdot |\mathcal{S}_{arv}|},$$

$$\bar{Q}(1,0) = \bar{Q}(4,3) = \bar{Q}(7,6) = \bar{Q}(9,8) = 0.5 I_{|\bar{\mathcal{S}}_{rtl}| \cdot |\mathcal{S}_{arv}|},$$

$$\bar{Q}(5,3) = 6 I_{|\bar{\mathcal{S}}_{rtl}| \cdot |\mathcal{S}_{arv}|},$$

$$\bar{Q}(9,6) = 2 I_{|\bar{\mathcal{S}}_{rtl}| \cdot |\mathcal{S}_{arv}|}.$$

Chapter 5
Vector–Kronecker Product Multiplication

The basic operation underlying iterative analysis of multidimensional CTMCs with generator matrices represented using sums of Kronecker products is vector–Kronecker product multiplication. Therein, the challenge is to perform this operation in as little of memory and as fast as possible.

When the matrices in the Kronecker product terms of (2.10) are relatively dense, vector–Kronecker product multiplication can be performed efficiently by the shuffle algorithm [136, 137, 263, 265, 266] discussed in the next section. When the matrices are relatively sparse, it may be more efficient to obtain nonzero entries of the generator matrix in Kronecker form on the fly and multiply them with corresponding entries of the vector [57]. Recently, [110] has suggested a modification to the shuffle algorithm that multiplies relevant entries of the vector with submatrices of matrices in which zero rows and columns are omitted. This approach, which is discussed after the shuffle algorithm, avoids unnecessary flops that evaluate to zero during the course of the multiplication and has the possibility to reduce the amount of memory used. In many cases, the modified shuffle algorithm yields a smaller number of flops than the shuffle algorithm and the algorithm that generates nonzeros on the fly, sometimes with a minimum number of flops and as little of memory possible. Unfortunately, the memory allocated for vectors in all mentioned algorithms is still proportional to the size of the reachable state space, and this size increases rapidly with the number of dimensions.

Recent advances in storing and analyzing dense multidimensional data numerically [177] have hinted at an approach to cope with the problem pertaining to the size of vectors used in vector–Kronecker product multiplication. In this approach, essential information in the data of a full vector with size equal to that of the reachable state space is represented compactly using a special kind of Kronecker decomposition, so that, rather than the full vector, a number of shorter vectors are employed with a user controllable accuracy

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_5

on the decomposition [163, 178, 202, 203, 256]. Preliminary results regarding the merits of this approach for representing vectors compactly will also be discussed in this chapter following [64].

5.1 Shuffle Algorithm

The shuffle algorithm is at the heart of all iterative solvers for sums of Kronecker products. It shows how one can multiply a vector with a Kronecker product of H matrices. Here we present its left-oriented version suitable for systems of equations arising from MCs, where the matrices are rectangular matrices $X^{(h)} \in \mathbb{R}^{n_h \times m_h}$ for $h = 1, \ldots, H$.

The algorithm is based on the identity

$$\bigotimes_{h=1}^{H} X^{(h)} = \prod_{h=1}^{H} (I_{m_1} \otimes \cdots \otimes I_{m_{h-1}} \otimes X^{(h)} \otimes I_{n_{h+1}} \otimes \cdots \otimes I_{n_H}) \,,$$

or more simply [136]

$$\bigotimes_{h=1}^{H} X^{(h)} = \prod_{h=1}^{H} (I_{\prod_{l=1}^{h-1} m_l} \otimes X^{(h)} \otimes I_{\prod_{l=h+1}^{H} n_l}). \tag{5.1}$$

In order to see this, let us first write for the Kronecker product of two factors as in

$$X^{(1)} \otimes X^{(2)} = (X^{(1)} I_{m_1}) \otimes (I_{n_2} X^{(2)}) = (X^{(1)} \otimes I_{n_2})(I_{m_1} \otimes X^{(2)}) \,,$$

where the first equality follows from $X^{(1)} = X^{(1)} I_{m_1}$ since $X^{(1)} \in \mathbb{R}^{n_1 \times m_1}$ and $X^{(2)} = I_{n_2} X^{(2)}$ since $X^{(2)} \in \mathbb{R}^{n_2 \times m_2}$ and the second equality follows from the compatibility of Kronecker product with matrix multiplication.

Let us now introduce a third factor to the Kronecker product as in

$$\begin{aligned} X^{(1)} \otimes X^{(2)} \otimes X^{(3)} &= (X^{(1)} \otimes I_{n_2})(I_{m_1} \otimes X^{(2)}) \otimes (I_{n_3} X^{(3)}) \\ &= (X^{(1)} \otimes I_{n_2} \otimes I_{n_3})(I_{m_1} \otimes X^{(2)} \otimes X^{(3)}) \,, \end{aligned}$$

where the first equality follows from the result for the Kronecker product of two factors and $X^{(3)} = I_{n_3} X^{(3)}$ since $X^{(3)} \in \mathbb{R}^{n_3 \times m_3}$ and the second equality follows from the compatibility of Kronecker product with matrix multiplication. Observe that this last equality can be rewritten as

$$\begin{aligned} &X^{(1)} \otimes X^{(2)} \otimes X^{(3)} = \\ &\quad (X^{(1)} \otimes I_{n_2} \otimes I_{n_3}) \left((I_{m_1} I_{m_1}) \otimes ((X^{(2)} \otimes I_{n_3})(I_{m_2} \otimes X^{(3)})) \right) . \end{aligned}$$

using the fact that $I_{m_1} = I_{m_1} I_{m_1}$ and the above result for the Kronecker product of two factors with $X^{(2)}$ and $X^{(3)}$. Then, again using the compati-

bility of Kronecker product with matrix multiplication, we obtain

$$X^{(1)} \otimes X^{(2)} \otimes X^{(3)} = (X^{(1)} \otimes I_{n_2} \otimes I_{n_3})(I_{m_1} \otimes X^{(2)} \otimes I_{n_3})(I_{m_1} \otimes I_{m_2} \otimes X^{(3)}).$$

Clearly, this approach can be generalized to more than three factors.

Now, note that $I_{\prod_{l=1}^{h-1} m_l} \otimes X^{(h)} \otimes I_{\prod_{l=h+1}^{H} n_l}$ is an identity matrix when $X^{(h)} = I_{n_h}$. Hence, the left multiplication of $\mathbf{x} \in \mathbb{R}^{1 \times \prod_{h=1}^{H} n_h}$ with $\bigotimes_{h=1}^{H} X^{(h)}$ can be accomplished as in Algorithm 4 yielding a product vector whose length ranges from $m_1 \prod_{h=2}^{H} n_h$ to $\prod_{h=1}^{H} m_h$ during the course of the multiplication.

ALGORITHM 4. *Shuffle algorithm for* $\mathbf{x}' := \mathbf{x} \bigotimes_{h=1}^{H} X^{(h)}$.

Copy $\mathbf{x}$ to $\mathbf{x}'$; $i_{left} := 1$; $i_{right} := \prod_{h=2}^{H} n_h$; $n_{H+1} := 1$;
For $h := 1$ to H,
  If $X^{(h)} \neq I_{n_h}$,
    $base_i := 0$; $base_j := 0$;
    For $i_l := 0, \ldots, i_{left} - 1$,
      For $i_r := 0, \ldots, i_{right} - 1$,
        $index_i := base_i + i_r$;
        For $row := 0, \ldots, n_h - 1$,
          $\mathbf{z}(row) := \mathbf{x}'(index_i)$; $index_i := index_i + i_{right}$;
        $\mathbf{z}' := \mathbf{z}X^{(h)}$;
        $index_j := base_j + i_r$;
        For $col := 0, \ldots, m_h - 1$,
          $\mathbf{x}''(index_j) := \mathbf{z}'(col)$; $index_j := index_j + i_{right}$;
      $base_i := base_i + n_h i_{right}$; $base_j := base_j + m_h i_{right}$;
    Copy $\mathbf{x}''$ to $\mathbf{x}'$;
  $i_{left} := i_{left} m_h$; $i_{right} := i_{right}/n_{h+1}$.

In (5.1), the hth matrix of the form $I_{\prod_{l=1}^{h-1} m_l} \otimes X^{(h)} \otimes I_{\prod_{l=h+1}^{H} n_l}$ is a rectangular $(\prod_{l=1}^{h-1} m_l \prod_{l=h}^{H} n_l \times \prod_{l=1}^{h} m_l \prod_{l=h+1}^{H} n_l)$ block diagonal matrix having $\prod_{l=1}^{h-1} m_l$ diagonal blocks each of size $(n_h \prod_{l=h+1}^{H} n_l \times m_h \prod_{l=h+1}^{H} n_l)$. Furthermore, each of the blocks along the diagonal is an $(n_h \times m_h)$ block matrix, where each subblock is a diagonal matrix of order $\prod_{l=h+1}^{H} n_l$ with a particular entry of $X^{(h)}$ appearing along its diagonal $\prod_{l=h+1}^{H} n_l$ many times. It is this feature that is used in devising the vector–Kronecker product multiplication algorithm (cf. [136]).

Observe that the only flops in Algorithm 4 take place in the vector–matrix multiplication $\mathbf{z}' = \mathbf{z}X^{(h)}$ when $X^{(h)} \neq I_{n_h}$, which can be simplified further if $X^{(h)} = I_{n_h \times m_h}$. The rest of the operations are index manipulation and copying of vector entries. For a fixed value of h, the multiplication is executed $\prod_{l=1}^{h-1} m_l \prod_{l=h+1}^{H} n_l$ times and the cost of a vector multiplication with $\bigotimes_{h=1}^{H} X^{(h)}$ amounts to

$$2 \sum_{h \in \mathcal{H}} nz_{X^{(h)}} \prod_{l=1}^{h-1} m_l \prod_{l=h+1}^{H} n_l \quad \text{flops},$$

where

$$\mathcal{H} = \{h \in \{1, \ldots, H\} \mid X^{(h)} \neq I_{n_h}\}$$

and $nz_{X^{(h)}}$ is the number of nonzeros in $X^{(h)}$ for $h \in \mathcal{H}$. When $\mathcal{H}$ is a singleton, we have $n_h = m_h = nz_{X^{(h)}}$ for $h \notin \mathcal{H}$ and $h = 1, \ldots, H$, and therefore, the number of flops executed by Algorithm 4 becomes $2nz_X$, where nz_X is the number of nonzeros in $X = \bigotimes_{h=1}^{H} X^{(h)}$.

Regarding space requirements, in addition to the input vector $\mathbf{x}$, the algorithm requires two temporary floating-point vectors, $\mathbf{z}$ and $\mathbf{z}'$, which need to be of length $\max_h(n_h)$ and $\max_h(m_h)$, respectively, and two floating-point vectors, $\mathbf{x}'$ and $\mathbf{x}''$, of length $\max_{h \in \{0\} \cup \mathcal{H}}(\prod_{l=1}^{h} m_l \prod_{l=h+1}^{H} n_l)$ to compute and return the result. When $X^{(h)} = I_{n_h \times m_h}$, entries of $\mathbf{x}'$ can be directly copied to $\mathbf{x}''$ using appropriate index manipulations. Furthermore, in programming languages which provide handles to memory addresses in the form of pointers, it is possible to copy $\mathbf{x}''$ to $\mathbf{x}'$ by swapping the two pointers to the arrays $\mathbf{x}'$ and $\mathbf{x}''$, since $\mathbf{x}''$ is to be rewritten in the next turn of the outer loop anyway. However, when the input vector needs to be multiplied with sums of Kronecker product terms as in (2.10), the output vector $\mathbf{x}'$ cannot be used to store intermediate results.

The cost of a vector multiplication with $\sum_{k=1}^{K} \bigotimes_{h=1}^{H} X_k^{(h)}$, where $X_k^{(h)} \in \mathbb{R}^{n_h \times m_h}$ for $h = 1, \ldots, H$ and

$$\mathcal{H}_k = \{h \in \{1, \ldots, H\} \mid X_k^{(h)} \neq I_{n_h}\} \quad \text{for } k = 1, \ldots, K,$$

is therefore

$$K \prod_{h=1}^{H} m_h + 2 \sum_{k=1}^{K} \sum_{h \in \mathcal{H}_k} nz_{X_k^{(h)}} \prod_{l=1}^{h-1} m_l \prod_{l=h+1}^{H} n_l \, .$$

Here, the first term is due to the summation of the K product vectors obtained through Algorithm 4. Note that this expression can be simplified further when the factors $X_k^{(h)}$ are square matrices by substituting n_h for m_h and using $n = \prod_{h=1}^{H} n_h$. In that case, one obtains $n(K + 2\sum_{k=1}^{K} \sum_{h \in \mathcal{H}'_k} nz_{X_k^{(h)}}/n_h)$, where

$$\mathcal{H}'_k = \{h \in \{1, \ldots, H\} \mid X_k^{(h)} \neq I_{n_h}\}$$

and $nz_{X_k^{(h)}}$ is the number of nonzeros in $X_k^{(h)}$ for $k = 1, \ldots, K$ and $h \in \mathcal{H}'_k$.

The next example is given to show how vector–Kronecker product multiplication works when Algorithm 4 is used. An example with rectangular matrices can be found in [110].

Example 1. (ctnd.) Consider the multiplication of the vector

$$\mathbf{x} = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17})$$

with the fourth Kronecker product term in our small example. That is, let $X = X^{(1)} \otimes X^{(2)} \otimes X^{(3)}$, where

$$X^{(1)} = \begin{pmatrix} & & \\ & & \\ \mu & & \end{pmatrix}_{3\times 3}, \; X^{(2)} = \begin{pmatrix} 1 & \\ & \end{pmatrix}_{2\times 2}, \; X^{(3)} = \begin{pmatrix} & & \\ & & \\ 1 & & \end{pmatrix}_{3\times 3},$$

and consider $\mathbf{x}' = \mathbf{x} \bigotimes_{h=1}^{3} X^{(h)}$. In Algorithm 4, the outer loop is executed three times for $h = 1, 2, 3$. At the beginning, $\mathbf{x}$ is copied to $\mathbf{x}'$, and $\mathbf{x}'' = \mathbf{x}'(X^{(1)} \otimes I_6)$ is computed for $h = 1$. In this turn, $nz_{X^{(1)}} = 1$, $i_{left} = 1$, and $i_{right} = 6$. Therefore, the second loop and the third loop are executed once and six times, respectively. Then $\mathbf{x}'(X^{(1)} \otimes I_6)$ is computed as

$$\mathbf{x}'' = (\mu x_{12}, \mu x_{13}, \mu x_{14}, \mu x_{15}, \mu x_{16}, \mu x_{17}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

in 12 flops, and $\mathbf{x}''$ is copied to $\mathbf{x}'$ at the end of the first turn. In the second turn, $\mathbf{x}'' = \mathbf{x}'(I_3 \otimes X^{(2)} \otimes I_3)$ is computed for $h = 2$. In this turn, $nz_{X^{(2)}} = 1$, $i_{left} = 3$, and $i_{right} = 3$. Therefore, the second loop and the third loop are each executed thrice. Then $\mathbf{x}'(I_3 \otimes X^{(2)} \otimes I_3)$ is computed as

$$\mathbf{x}'' = (\mu x_{15}, \mu x_{16}, \mu x_{17}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

in 18 flops, and $\mathbf{x}''$ is copied to $\mathbf{x}'$ at the end of the second turn. In the last turn, $\mathbf{x}'' = \mathbf{x}'(I_6 \otimes X^{(3)})$ is computed for $h = 3$. In this turn, $nz_{X^{(3)}} = 1$, $i_{left} = 6$, and $i_{right} = 1$. Therefore, the second loop and the third loop are executed six times and once, respectively. Then $\mathbf{x}'(I_6 \otimes X^{(3)})$ is computed as

$$\mathbf{x}'' = (\mu x_{17}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

in 12 flops, and $\mathbf{x}''$ is copied back to $\mathbf{x}'$ at the end of the last turn. Hence, Algorithm 4's computation of $\mathbf{x}' = \mathbf{x}X$ takes altogether 42 flops.

As mentioned before, functional transitions are a feature of SANs and are said to generalize the Kronecker product. Functional transitions enable the modeling of dependencies among subsystems elegantly. Yet, they are not easy to implement efficiently. There is a vector-generalized Kronecker product multiplication algorithm [136] which may be used in the presence of functional transitions. For instance, in the PEPS tool where it is implemented, the different values the functions can take are hard coded into the package. And this is done for each problem under consideration. Whenever a function needs to be evaluated in a particular state during the vector-generalized Kronecker product multiplication, the code jumps to a specific location dictated by the problem and carries out a sequence of if statements, trying to find out what the function evaluates to for that state. Although certain improvements have been introduced regarding this aspect in the last version of the PEPS tool [138], in practice this feature slows down the code since computation-intensive operations are interleaved with other operations, but does not come across when one looks at the time complexity result of vector-generalized Kronecker product multiplication.

The pseudocode of the implementation of the first loop of Algorithm 4 in the `Nsolve` package of the APNN toolbox [7, 22] is given in [43]. Identity matrices are not stored in this implementation as discussed in [43], and the vector–matrix multiplication is not performed for identity matrices. Note that the shuffle algorithm cannot utilize the cache efficiently when the value of index h in the outer loop of Algorithm 4 becomes large.

In order to facilitate a fair comparison among different vector–Kronecker product multiplication algorithms, we provide an implementation in [115]. Through this implementation, we observe that in addition to identity matrices, it is useful not to store matrices that are multiples of identity matrices. We also find it to be more efficient to multiply each nonzero of $X^{(h)}$ with the entries of the input vector and add the result to appropriate entries of the output vector so that $\mathbf{z}$ and $\mathbf{z}'$ are not used and stored in vector–matrix multiplication. Naturally, the models with which we experiment have more than one Kronecker product term, and therefore, intermediate results cannot be stored in the output vector. However, the number of auxiliary vectors needed to store intermediate results is two only if the maximum number of nonidentity matrices across all Kronecker product terms is more than two. Otherwise, one auxiliary vector is sufficient if the maximum number of nonidentity matrices across all terms is two, and no auxiliary vector is needed if this number is one since there is no intermediate result in that case. However, it should be remarked that at least two auxiliary vectors of length $\max_p |\mathcal{R}_p|$ are required in iterative solvers.

If the Q matrix corresponding to a model was generated and it had nz nonzeros in its off-diagonal part, then $2nz$ flops would be performed when a vector of length $|\mathcal{R}|$ is multiplied with the off-diagonal part of Q. Therefore, $2nz$ flops is a lower bound for the case of relatively sparse Kronecker product matrices, and one should be happy to see flop counts that are close to $2nz$ with vector–Kronecker product multiplication algorithms.

5.2 Modified Shuffle Algorithm

In the shuffle algorithm, the number of flops executed due to a particular matrix does not depend on the nonzero structure of the other matrices in the Kronecker product. However, when other matrices include zero rows or columns, some vector entries end up being computed even though they would evaluate to zero at the end. Recently in [110], the shuffle algorithm is improved by omitting zero rows and columns in matrices of Kronecker product terms during multiplication. In this way, flops evaluating to zero during the course of the multiplication are avoided, and possibly the memory requirements are reduced. Although the idea may seem simple, it is not intuitive since zero rows and columns are not stored in sparse matrices anyway. But, as we shall see, it turns out to be quite effective in many cases.

Now, let $\mathcal{N}^{(h)} \subseteq \{0,\ldots,n_h-1\}$ and $\mathcal{M}^{(h)} \subseteq \{0,\ldots,m_h-1\}$ denote the sets of rows and columns of $X^{(h)} \in \mathbb{R}^{n_h \times m_h}$ that include at least one nonzero for $h = 1,\ldots,H$. Furthermore, let $P_{(u,\mathcal{U})} \in \{0,1\}^{u\times|\mathcal{U}|}$ be the matrix given entrywise as

$$P_{(u,\mathcal{U})}(i,j) \begin{cases} 1 & \text{if } \ j = f^{(\mathcal{U})}(i) \text{ and } i \in \mathcal{U} \\ 0 & \text{otherwise} \end{cases},$$

where u is a finite positive integer, $\mathcal{U} \subseteq \{0,\ldots,u-1\}$ is a nonempty set, and

$$f^{(\mathcal{U})}(i) = |\{j \in \mathcal{U} \mid j < i\}| \quad \text{for } i \in \mathcal{U}.$$

Then

$$\bigotimes_{h=1}^{H} X^{(h)} = \left(\bigotimes_{h=1}^{H} P_{(n_h,\mathcal{N}^{(h)})}\right) \bigotimes_{h=1}^{H} \hat{X}^{(h)} \left(\bigotimes_{h=1}^{H} P^T_{(m_h,\mathcal{M}^{(h)})}\right), \tag{5.2}$$

where

$$\hat{X}^{(h)} = P^T_{(n_h,\mathcal{N}^{(h)})}\, X^{(h)}\, P_{(m_h,\mathcal{M}^{(h)})} \quad \text{for } h = 1,\ldots,H.$$

The proof in [110] rests on forming the $(u \times u)$ diagonal matrix $\bar{P}_{(u,\mathcal{U})} = P_{(u,\mathcal{U})}P^T_{(u,\mathcal{U})}$, showing that $X^{(h)}$ is invariant under pre-multiplication by $\bar{P}_{(n_h,\mathcal{N}^{(h)})}$ and post-multiplication by $\bar{P}_{(m_h,\mathcal{M}^{(h)})}$ and using the compatibility of Kronecker product with matrix multiplication.

As a corollary of this result, we obtain the identity [110]

$$\mathbf{x}'(\bigtimes_{h=1}^{H} \mathcal{M}^{(h)}) = \mathbf{x}(\bigtimes_{h=1}^{H} \mathcal{N}^{(h)}) \bigotimes_{h=1}^{H} \hat{X}^{(h)},$$

which follows from $\mathbf{x}' = \mathbf{x} \bigotimes_{h=1}^{H} X^{(h)}$,

$$\mathbf{x}(\bigtimes_{h=1}^{H} \mathcal{N}^{(h)}) = \mathbf{x} \bigotimes_{h=1}^{H} P_{(n_h,\mathcal{N}^{(h)})}, \quad \mathbf{x}'(\bigtimes_{h=1}^{H} \mathcal{M}^{(h)}) = \mathbf{x}' \bigotimes_{h=1}^{H} P_{(m_h,\mathcal{M}^{(h)})},$$

and

$$P^T_{(m_h,\mathcal{M}^{(h)})} P_{(m_h,\mathcal{M}^{(h)})} = I_{|\mathcal{M}^{(h)}|} \quad \text{for } h = 1,\ldots,H.$$

It is this identity on which the modified shuffle algorithm is based.

The modified shuffle algorithm given in Algorithm 5 executes flops only while multiplying vector $\hat{\mathbf{x}}$ with $\bigotimes_{h=1}^{H} \hat{X}^{(h)}$. Hence, the cost of vector–Kronecker product multiplication using the modified version amounts to

$$2 \sum_{h\in\mathcal{H}} nz_{X^{(h)}} \prod_{l=1}^{h-1} |\mathcal{M}^{(l)}| \prod_{l=h+1}^{H} |\mathcal{N}^{(l)}|$$

flops since $nz_{\hat{X}^{(h)}} = nz_{X^{(h)}}$ for $h = 1,\ldots,H$. Note that this never exceeds the cost of Algorithm 4 and is bounded above by $2\,|\mathcal{H}|\; nz_X$ flops since $|\mathcal{N}^{(h)}| \leqslant nz_{X^{(h)}}$ and $|\mathcal{M}^{(h)}| \leqslant nz_{X^{(h)}}$ for $h = 1,\ldots,H$.

ALGORITHM 5. *Modified shuffle algorithm for* $\mathbf{x}' := \mathbf{x} \bigotimes_{h=1}^{H} X^{(h)}$.

For $h := 1$ to H,
  $\mathcal{N}^{(h)} := \varnothing$; $\mathcal{M}^{(h)} := \varnothing$;
  For (i_h, j_h) such that $X^{(h)}(i_h, j_h) > 0$,
    $\mathcal{N}^{(h)} := \mathcal{N}^{(h)} \cup \{i_h\}$; $\mathcal{M}^{(h)} := \mathcal{M}^{(h)} \cup \{j_h\}$;
  Copy $X^{(h)}(\mathcal{N}^{(h)}, \mathcal{M}^{(h)})$ to $\hat{X}^{(h)}$;
Copy $\mathbf{x}(\times_{h=1}^{H} \mathcal{R}^{(h)})$ to $\hat{\mathbf{x}}$;
Execute shuffle algorithm for $\hat{\mathbf{x}}' := \hat{\mathbf{x}} \bigotimes_{h=1}^{H} \hat{X}^{(h)}$;
Copy $\hat{\mathbf{x}}'$ to $\mathbf{x}'(\times_{h=1}^{H} \mathcal{M}^{(h)})$.

Regarding memory requirements, each of the vectors $\mathbf{x}'$ and $\mathbf{x}''$ in Algorithm 5 needs to be of length $\max_{h \in \{0\} \cup \mathcal{H}}(\prod_{l=1}^{h} |\mathcal{M}^{(l)}| \prod_{l=h+1}^{H} |\mathcal{N}^{(l)}|)$, whereas the floating-point vectors $\mathbf{z}$ and $\mathbf{z}'$ need to be at least of length $\max_h(|\mathcal{N}^{(h)}|)$ and $\max_h(|\mathcal{M}^{(h)}|)$, respectively.

We use our running example to show how the modified shuffle algorithm works.

Example 1. (cntd.) The sets of rows and columns of the matrices $X^{(1)}$, $X^{(2)}$, and $X^{(3)}$ including at least one nonzero value are given by Algorithm 5 as

$$\mathcal{N}^{(1)} = \{2\},\ \mathcal{M}^{(1)} = \{0\},\ \mathcal{N}^{(2)} = \{1\},\ \mathcal{M}^{(2)} = \{0\},\ \mathcal{N}^{(3)} = \{2\},\ \mathcal{M}^{(3)} = \{0\}.$$

Then

$$\bigtimes_{h=1}^{3} \mathcal{N}^{(h)} = \{(2,1,2)\}, \quad \bigtimes_{h=1}^{3} \mathcal{M}^{(h)} = \{(0,0,0)\},$$

$$P_{(n_1,\mathcal{N}^{(1)})} = \begin{pmatrix} \\ 1 \end{pmatrix}_{3\times 1}, \quad P_{(m_1,\mathcal{M}^{(1)})} = \begin{pmatrix} 1 \\ \end{pmatrix}_{3\times 1}, \quad P_{(n_2,\mathcal{N}^{(2)})} = \begin{pmatrix} 1 \end{pmatrix}_{2\times 1},$$

$$P_{(m_2,\mathcal{M}^{(2)})} = \begin{pmatrix} 1 \end{pmatrix}_{2\times 1}, \quad P_{(n_3,\mathcal{N}^{(3)})} = \begin{pmatrix} \\ 1 \end{pmatrix}_{3\times 1}, \quad P_{(m_3,\mathcal{M}^{(3)})} = \begin{pmatrix} 1 \\ \end{pmatrix}_{3\times 1},$$

$$\bigotimes_{h=1}^{3} P_{(n_h,\mathcal{N}^{(h)})} = (\mathbf{e}_{17})_{18\times 1}, \quad \text{and} \quad \bigotimes_{h=1}^{3} P_{(m_h,\mathcal{M}^{(h)})} = (\mathbf{e}_0)_{18\times 1}.$$

Hence,

$$\hat{\mathbf{x}} = \mathbf{x} \bigotimes_{h=1}^{3} P_{(n_h,\mathcal{N}^{(h)})} = \mathbf{x}(\bigtimes_{h=1}^{3} \mathcal{N}^{(h)}) = (x_{17}).$$

Then $\hat{\mathbf{x}}' = \hat{\mathbf{x}} \otimes_{h=1}^{H} \hat{X}^{(h)}$ is computed using the shuffle algorithm, where

$$\hat{X}^{(1)} = \begin{pmatrix} \mu \end{pmatrix}_{1\times 1}, \quad \hat{X}^{(2)} = \begin{pmatrix} 1 \end{pmatrix}_{1\times 1}, \quad \text{and} \quad \hat{X}^{(3)} = \begin{pmatrix} 1 \end{pmatrix}_{1\times 1}.$$

In Algorithm 4, the outer loop is executed three times for $h = 1, 2, 3$. At the beginning, $\hat{\mathbf{x}}$ is copied to $\mathbf{x}'$ and $\mathbf{x}'' = \mathbf{x}'\ (\hat{X}^{(1)} \otimes I_1)$ is computed for $h = 1$. In this turn, $nz_{\hat{X}^{(1)}} = 1$, $i_{left} = 1$, and $i_{right} = 1$. Therefore, the second loop and the third loop are each executed once. Then $\mathbf{x}'\ (\hat{X}^{(1)} \otimes I_1)$ is computed as

$$\mathbf{x}'' = (\mu x_{17})$$

in 2 flops, and $\mathbf{x}''$ is copied to $\mathbf{x}'$ at the end of the first turn. In the second turn, $\mathbf{x}'' = \mathbf{x}'\ (I_1 \otimes \hat{X}^{(2)} \otimes I_1)$ is computed for $h = 2$. In this turn, $nz_{\hat{X}^{(2)}} = 1$, $i_{left} = 1$, and $i_{right} = 1$. Therefore, the second loop and the third loop are each executed once. Then $\mathbf{x}'\ (I_1 \otimes \hat{X}^{(2)} \otimes I_1)$ is computed as

$$\mathbf{x}'' = (\mu x_{17})$$

in 2 flops, and $\mathbf{x}''$ is copied to $\mathbf{x}'$ at the end of the second turn. In the last turn, $\mathbf{x}'' = \mathbf{x}'\ (I_1 \otimes \hat{X}^{(3)})$ is computed for $h = 3$. In this turn, $nz_{\hat{X}^{(3)}} = 1$, $i_{left} = 1$, and $i_{right} = 1$. Therefore, the second loop and the third loop are each executed once. Then $\mathbf{x}'\ (I_1 \otimes \hat{X}^{(3)})$ is computed as

$$\mathbf{x}'' = (\mu x_{17})$$

in 2 flops, and $\mathbf{x}''$ is copied to $\hat{\mathbf{x}}'$ at the end of the last turn. Then the entries of $\hat{\mathbf{x}}'$ are copied back to $\mathbf{x}'(\times_{h=1}^{3} \mathcal{M}^{(h)})$, that is,

$$\mathbf{x}' = (\mu x_{17}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

by the equation

$$\hat{\mathbf{x}}' = \mathbf{x}' \bigotimes_{h=1}^{3} P_{(m_h, \mathcal{M}^{(h)})} = \mathbf{x}'(\bigtimes_{h=1}^{3} \mathcal{M}^{(h)}).$$

Hence, Algorithm 5's computation of $\hat{\mathbf{x}}' = \hat{\mathbf{x}} \bigotimes_{h=1}^{3} \hat{X}^{(h)}$ takes altogether six flops. This number is smaller than the 42 flops which the shuffle algorithm performs.

The nice thing about the modified shuffle algorithm is that any improvement it brings in the number of flops can be predetermined before it is run. This is also true for any improvement in memory. This algorithm is also likely to be useful when the matrices that form Kronecker products are relatively dense as long as some of them include zero rows or columns.

The improvement in time obtained with a vector–Kronecker product multiplication algorithm depends on the particular matrices in the Kronecker product terms. Besides the number of flops, the time that the algorithm takes also depends on the overhead of indexing and addressing as well as the access pattern to the input and output vectors. In this respect, time per flop seems to be a good measure to understand the overhead and cache usage of

different algorithms. The cost due to indexing, addressing, and access pattern needs to be smaller than the gain obtained from the decrease in the number of flops to observe speed up.

In the particular implementation provided in [115], zero rows and columns of nonidentity matrices are removed, and two integer vectors are allocated to store the mapping between row and column indices of the matrix and the modified matrix. This modification may avoid unnecessary flops only if the Kronecker product term includes more than one nonidentity matrix. Since choosing entries of input and output vectors yields indexing and addressing overhead in many cases, we remove zero rows and columns of nonidentity matrices in a Kronecker product term only if their counts are more than one. Furthermore, rather than copying appropriate entries of the input vector to a temporary vector, multiplying the temporary vector with the Kronecker product of the submatrices, and adding the resulting vector to the output vector, we modify the implementation of the shuffle algorithm so that appropriate entries of the input vector are chosen in the first turn of the outer loop of Algorithm 4 and the entries of the resulting vector are added to the appropriate entries of the output vector in the last turn of the outer loop of Algorithm 4.

Detailed results of numerical experiments with the models introduced earlier are provided in [110]. The average number of nonzeros per row in the nonidentity matrices of Kronecker product terms of the models is close to 1. This implies that the Kronecker product matrices of models are relatively sparse for the shuffle algorithm to be efficient. For each model, a smaller number of flops are observed with the modified shuffle algorithm. There are two reasons for this decrease. First, unnecessary flops in the outer loop, for some fixed h, in Algorithm 4 are avoided. Second, some matrices become multiples of identity matrices once zero rows and columns are removed, and the outer loop of Algorithm 4 is avoided for such matrices. The improvement in the number of flops is about 30% over all models considered. In more than half of the models, the modified shuffle algorithm performs $2nz$ flops, which is the lower bound. We remark that in each of these models, there is no term with more than one explicitly stored matrix. In the models where modified shuffle performs $2nz$ flops, it also does not require additional memory. Besides, the modification also decreases the memory allocated for auxiliary vectors over that of the shuffle algorithm in some other models. When the expected number of explicitly stored matrices becomes relatively larger than 1, the performance of modified shuffle deteriorates.

5.3 Working with Compact Solution Vectors

In this section, we discuss a recently proposed approach that may be used to reduce the amount of memory allocated to solution vectors in Kronecker-based Markovian analysis so that larger problems can be analyzed on the

same platform. The approach is based on using the *hierarchical Tucker decomposition* (HTD) [163, 178, 202, 203] to represent each solution vector at hand compactly by a number of shorter vectors, which happen to be the columns of particular matrices obtained through the decomposition, and Kronecker operations as discussed in [64]. Similar to the *tensor-train decomposition* [256], HTD is originally conceived in the context of providing a compact approximate representation for dense multidimensional data [177]. Although HTD is more suitable for our purposes since the decomposition is available through a binary tree data structure with logarithmic depth in the number of dimensions, both HTD and tensor-train decompositions have the desirable feature of a user controllable approximation error. This implies that potentially approximations accurate to machine precision can be computed through both decompositions, though this may not necessarily yield a reduction in memory requirements. Since compactness of the representation can be traded for quality of approximation when using such decompositions, how compact a solution vector in HTD format remains throughout the analysis process by controlling its accuracy is worthwhile exploring.

Compact representations for solution vectors in Markovian analysis are considered earlier from the perspective of binary decision diagrams in [69, 209]. However, the proposed compact structures therein are not time-wise competitive, and they do not allow the computation of approximation error bounds. On the other hand, the tensor-train decomposition is recently used in [201] to approximate the solution vector of communicating Markovian models in which the product state space is reachable through alternating least squares and power methods. The approach discussed in this section is geared toward generator matrices in Kronecker form with reachable state spaces possibly smaller than their product state spaces and seems to be a step forward [64].

As we will see, the initialization of a compact vector in HTD format at the outset with a rank-1 vector and the computation of the 2-norm of a compact vector in HTD format so that it may be used in the test for convergence in iterative analysis turn out to be straightforward operations. However, the multiplication of a compact vector in HTD format with a sum of Kronecker products that is used at each iteration poses a problem. It is not the multiplication of a compact vector in HTD format with a Kronecker product but the addition operation between two compact vectors in HTD format that increases the memory requirements of the sum. Unfortunately, this necessitates some kind of truncation, hence, approximation, to be introduced to the addition operation. So, let us first introduce the HTD format and discuss these issues in order. As before, we will be using row solution vectors $\mathbf{x}$ that multiply Q or any part of it in Kronecker form on the left.

The original *Tucker decomposition* [313] represents 3-dimensional data as the multiplication of 3-dimensional core (or compressed) data that is all-orthogonal (a concept corresponding to orthogonality across all dimensions) and an orthogonal basis matrix along each dimension. Higher-order singular value decomposition (HOSVD) [124] provides a simple and nearly optimal

solution to approximating multidimensional data in Tucker decomposition format with a user controllable accuracy. Since the memory requirement of multidimensional core data scales exponentially with the number of dimensions, a low-rank approximation of HOSVD is to be used in practice.

Assuming without loss of generality that H is a power of 2, the $(n \times 1)$ vector $\mathbf{x}^T$ in *(orthogonalized) HTD format* is expressed as

$$\mathbf{x}^T = (U^{(1)} \otimes \cdots \otimes U^{(H)})\mathbf{c},$$

where $U^{(h)} \in \mathbb{R}^{n_h \times r_h}$ is the *orthogonal basis matrix* associated with dimension h in the model for $h = 1, \ldots, H$ and

$$\mathbf{c} = (B^{(1,2)} \otimes \cdots \otimes B^{(H-1,H)}) \cdots (B^{(1,\ldots,H/2)} \otimes B^{(H/2+1,\ldots,H)})B^{(1,\ldots,H)}$$

is a $(\prod_{h=1}^{H} r_h \times 1)$ vector in the form of a product of exactly $\lg H$ matrices each of which except the last is a Kronecker product of a number of *transfer matrices* $B^{(\bar{t})}$ related to each other as in the full binary tree of Figure 5.1.

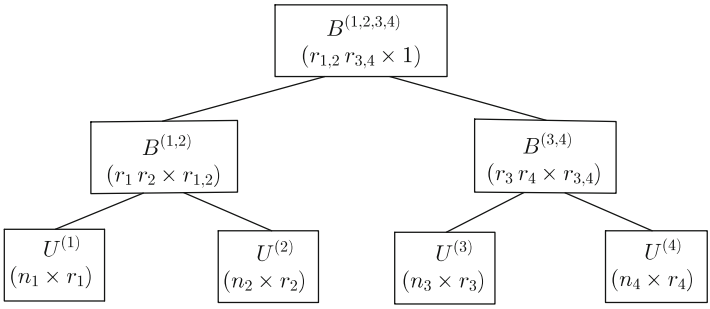


Fig. 5.1 Matrices forming $\mathbf{x}^T$ in HTD format for $H = 4$.

The transfer matrix $B^{(\bar{t})}$ is of size $(r_{\bar{t}_l} r_{\bar{t}_r} \times r_{\bar{t}})$ with the node index $\bar{t}$ defined as $\bar{t} := \bar{t}_l, \bar{t}_r$, and $r_{1,\ldots,H} = 1$ since $B^{(1,\ldots,H)}$ is at the root of the tree. The $(H-1)$ intermediate nodes of the binary tree including the root in Figure 5.1 store the transfer matrices $B^{(\bar{t})}$, and its leaves store the basis matrices $U^{(h)}$ so that each intermediate node has two children.

In orthogonalized HTD format of $\mathbf{x}^T$, one can also conceive of orthogonal basis matrices

$$U^{(\bar{t})} = (U^{(\bar{t}_l)} \otimes U^{(\bar{t}_r)})B^{(\bar{t})},$$

at intermediate nodes with $r_{\bar{t}}$ columns that relate the orthogonal basis matrices $U^{(\bar{t}_l)}$ and $U^{(\bar{t}_r)}$ for the two children of transfer matrix $B_{\bar{t}}$ with the transfer matrix itself. Now, recall that the *singular values* [125, 158, 303] of a real rectangular matrix X are the square roots of the eigenvalues of $X^T X$, which by construction is symmetric positive definite and therefore has positive real eigenvalues. The orthogonal matrix $U^{(\bar{t})}$ has in its columns the *singular vec-*

tors associated with the largest $r_{\bar{t}}$ singular values of the matrix obtained by taking index $\bar{t}$ as row index, the remaining indices in order as column index of the H-dimensional data at hand, that is, $\mathbf{x}(\{\bar{t}\}, \{1, \ldots, H\} \backslash \{\bar{t}\}))$. Hence, we have a hierarchy of *matricizations* and HOSVD, where $r_{\bar{t}}$ is the rank of the *truncated HOSVD*. The hope is to be able to obtain a relatively accurate decomposition in which $r_{\bar{t}}$'s are small. More information regarding this can be found in [163, 177, 178, 202].

We remark that $B^{(\bar{t})}$ may also be perceived as a 3-dimensional array of size $(r_{\bar{t}_l} \times r_{\bar{t}_r} \times r_{\bar{t}})$ having as many indices in each of its three dimensions as the number of columns in the matrices of its two children and itself, respectively. The number of transfer matrices in the lth factor forming $\mathbf{c}$ is the Kronecker product of $2^{\lg H - l}$ transfer matrices for $l = 1, \ldots, \lg H - 1$. In fact, $\mathbf{c}$ is a product of Kronecker products, and so is $\mathbf{x}^T$, but neither is formed explicitly.

When H is not a power of 2, it is still important to keep the tree in a balanced form, for instance, as in Figure 5.2 for which

$$\mathbf{x}^T = (U^{(1)} \otimes U^{(2)} \otimes U^{(3)} \otimes U^{(4)} \otimes U^{(5)} \otimes U^{(6)} \otimes U^{(7)})\mathbf{c}$$

and

$$\mathbf{c} = (B^{(1,2)} \otimes B^{(3,4)} \otimes B^{(5,6)} \otimes I_{r_7})(B^{(1,2,3,4)} \otimes B^{(5,6,7)})B^{(1,2,3,4,5,6,7)}$$

so that the height of the tree does not exceed $\lceil \lg H \rceil$.

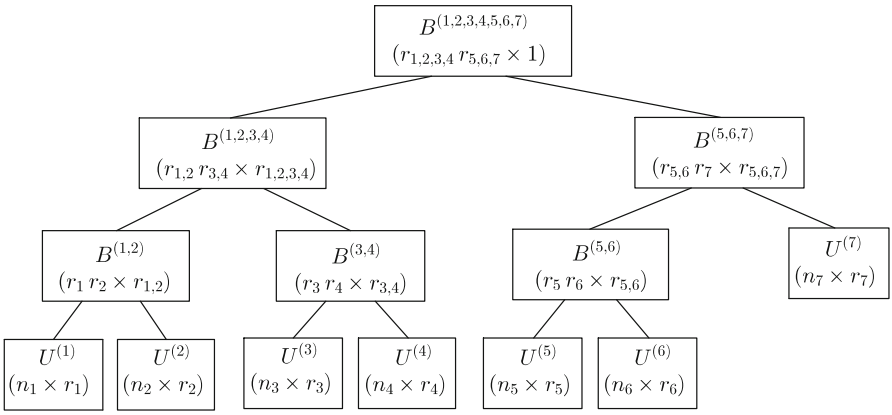


Fig. 5.2 Matrices forming $\mathbf{x}^T$ in HTD format for $H = 7$.

Assuming that $r_{\max} = \max_{\bar{t}}(r_{\bar{t}})$ and $n_{\max} = \max(n_1, \ldots, n_H)$, memory requirement for matrices in the binary tree associated with HTD format is bounded by $Hn_{\max}r_{\max}$ at the leaves, $r_{\max}^2$ at the root, and $(H-2)r_{\max}^3$ at other intermediate nodes, thus totally $Hn_{\max}r_{\max} + (H-2)r_{\max}^3 + r_{\max}^2$.

Unless something else is known about the solution itself, the initial solution vector used with iterative methods in Markovian analysis is almost always the uniform probability distribution. Next we show how this particular rank-1 vector can be represented in HTD format.

Example 6. Let $\mathbf{x}^T = \mathbf{e}/n \in \mathbb{R}^{n\times 1}_{>0}$ be the uniform distribution vector, where $n = \prod_{h=1}^{H} n_h$. Then $\mathbf{x}^T$ may be represented in HTD format with all matrices having rank-1 for which the basis matrices given by $U^{(h)} = \mathbf{e}/\sqrt{n_h}$ are of size $(n_h \times 1)$ for $h = 1, \ldots, H$ and the transfer matrices given by

$$B^{(\bar{t})} = \begin{cases} (\prod_{h=1}^{H} \sqrt{n_h})/n & \text{if } \bar{t} \text{ corresponds to root} \\ 1 & \text{otherwise} \end{cases}$$

are (1×1). Note that memory taken up by the full-vector representation of $\mathbf{x}$ is n nonzeros whereas that with HTD format is $H - 1 + \sum_{h=1}^{H} n_h$ nonzeros since the $(H - 1)$ transfer matrices are all scalars equal to 1 except the one corresponding to the root. We remark that each basis matrix $U^{(h)}$ for the uniform distribution has only a single column and that column is unit 2–norm, implying all $U^{(h)}$ are orthogonal.

Now, let us move to the multiplication of a compact vector in HTD format with a sum of Kronecker products. We consider the multiplication of block $Q(p, w)$ in (2.10) from the left with a compact (sub)vector. Hence, we are interested in the operation

$$\mathbf{y} := \mathbf{x} \sum_{k=1}^{K} \bigotimes_{h=1}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)}),$$

where K is equal to the number of terms in the sum, that is, $|\mathcal{K}_{p,w}|$, for $Q(p, w)$ and $\mathbf{x}$ is in HTD format. Observe that this operation executes when each block of Q in (2.10) is pre-multiplied by an iteration subvector. In fact, the same subvector $\mathbf{x}$ pre-multiplies all blocks in block row p of the matrix in Kronecker form.

To simplify the notation, we rewrite the pre-multiplication with a sum of Kronecker products as

$$\mathbf{y} := \mathbf{x} \sum_{k=1}^{K} \bigotimes_{h=1}^{H} X_k^{(h)},$$

where $X_k^{(h)} \in \mathbb{R}^{n_h \times m_h}$, implying $\bigotimes_{h=1}^{H} X_k^{(h)} \in \mathbb{R}^{n\times m}$ and $\mathbf{x} \in \mathbb{R}^{1\times n}$. To be consistent with the form of HTD in the literature, we consider the following post-multiplications of Kronecker products $\otimes_{h=1}^{H} A_k^{(h)}$ with column vector $\mathbf{x}^T$ and their summation in the usual matrix–vector form

$$\mathbf{y}^T := \sum_{k=1}^{K} \left(\bigotimes_{h=1}^{H} A_k^{(h)} \right) \mathbf{x}^T,$$

where $A_k^{(h)} \in \mathbb{R}^{m_h \times n_h}$ is the transpose of $X_k^{(h)}$. In particular, we are interested in its implementation as

$$\mathbf{y}_{(1)}^T := (\mathbf{0})^T,$$

$$\mathbf{x}_{(k)}^T := \left(\bigotimes_{h=1}^{H} A_k^{(h)} \right) \mathbf{x}^T, \quad \mathbf{y}_{(k+1)}^T := \mathbf{y}_{(k)}^T + \mathbf{x}_{(k)}^T \quad \text{for } k := 1, \ldots, K,$$

$$\text{and} \quad \mathbf{y}^T := \mathbf{y}_{(K+1)}^T.$$

Assuming that $\mathbf{x}^T$ is in HTD format with orthogonal basis matrices $U^{(h)}$ and transfer matrices $B^{(\bar{t})}$ forming vector $\mathbf{c}$, the operation

$$\mathbf{x}_{(k)}^T := \left(\bigotimes_{h=1}^{H} A_k^{(h)} \right) \mathbf{x}^T \quad \text{amounts to computing}$$

$$\mathbf{x}_{(k)}^T := \left(\bigotimes_{h=1}^{H} A_k^{(h)} U^{(h)} \right) \mathbf{c} \quad \text{since} \quad \mathbf{x}^T = (\bigotimes_{h=1}^{H} U^{(h)})\mathbf{c}.$$

Hence, the only thing that needs to be done to carry out the computation of $\mathbf{x}_{(k)}^T$ in HTD format is to multiply the $(m_h \times n_h)$ Kronecker factor $A_k^{(h)}$ with the corresponding $(n_h \times r_h)$ orthogonal basis matrix $U^{(h)}$ for $h = 1, \ldots, H$. Clearly, the $(m_h \times r_h)$ product matrix $A_k^{(h)} U^{(h)}$ need not be orthogonal. But this does not pose much of a problem, since $\mathbf{x}_{(k)}^T$ can be transformed into orthogonalized HTD format if the need arises by computing the *QR decomposition* [125, 158, 302] of

$$A_k^{(h)} U^{(h)} = \tilde{U}^{(h)} R^{(h)} \quad \text{for } h = 1, \ldots, H,$$

propagating the triangular factors $R^{(h)}$ into the transfer matrices and orthogonalizing the updated transfer matrices at intermediate nodes in a similar manner up to the root [202]. However, the situation is not as good for the addition of two compact vectors.

Addition of two matrices Y and X with *singular value decompositions* (SVDs) [125, 158, 302]

$$Y = U_Y \Sigma_Y V_Y^T \quad \text{and} \quad X = U_X \Sigma_X V_X^T$$

results in

$$Y + X := (U_Y \; U_X) \begin{pmatrix} \Sigma_Y & \\ & \Sigma_X \end{pmatrix} (V_Y \; V_X)^T.$$

Here, Σ_Y, Σ_X are diagonal matrices of singular values, whereas U_Y, U_X and V_Y, V_X are orthogonal matrices of left and right singular (row) vectors associated with matrices Y and X, respectively. SVD is a rank revealing factorization in that the number of nonzero singular values of a matrix corresponds to its column rank. This implies that the sum $Y + X$ has a rank equal to the sum of the ranks of the two matrices that are added.

The situation for the sum $\mathbf{y}_{(k+1)}^T$ of the two vectors $\mathbf{y}_{(k)}^T$ and $\mathbf{x}_{(k)}^T$ in HTD format is no different if one replaces the SVD with HOSVD, which is illustrated for $H = 4$ in [202]. For the following steps performing the addition and representing the resulting vector in HTD format, we exploit the algorithms presented as part of the `htucker` toolbox [202]. Among three alternative approaches that have been investigated therein for computing $\mathbf{y}^T$, the best is to multiply, add, and then truncate K times without initial orthogonalization as argued asymptotically and demonstrated experimentally for larger K [202]. This approach works by performing the reduced Gramian computations of a compact vector in non-orthogonalized HTD format. Recall that the compact vector $\mathbf{x}_{(k)}^T$ obtained after multiplication does not need to be in orthogonal HTD format even though $\mathbf{x}^T$ might have been. Once the reduced Gramian computations of $\mathbf{x}_{(k)}^T$ are carried out, the truncated HOSVD for the sum of two vectors $\mathbf{y}_{(k)}^T$ and $\mathbf{x}_{(k)}^T$ in HTD format without initial orthogonalization can be computed. The output vector $\mathbf{y}_{(k+1)}^T$ is a truncated compact vector in orthogonalized HTD format, and this operation is repeated K times until $\mathbf{y}^T$ is obtained. The number of flops executed in this approach is $O(HK^2 r_{\max}^2 (m_{\max} + r_{\max}^2 + K r_{\max}))$, where $m_{\max} = \max(m_1, \ldots, m_H)$. The significance of this result is that one can impose an accuracy of `trunc_tol` on the truncated HOSVD by choosing rank $r_{\bar{t}}$ in node $\bar{t}$ based on dropping the smallest singular values whose squared sum is less than or equal to $\texttt{trunc\_tol}^2/(2H-3)$ [202]. This is a profound result but also implies that the truncation leads to an approximate solution vector. By setting `trunc_tol` to a small value, one is able to compute accurate solutions, which may sometimes imply increased memory requirements. On the other hand, it is possible to upper bound the memory requirements by prescribing a maximum value for the $r_{\bar{t}}$'s as in the `htucker` toolbox [202, 203]; but, this results in an a priori unknown approximation error.

In Markovian analysis, it is more relevant to compute the maximum (i.e., infinity) norm of a solution vector even though all norms are known to be equivalent [158, 302]. However, the computation of the maximum value in magnitude of the entries of a compact vector requires being able to locate the value and its index, which seems to be costly for a compact vector in HTD format. Therefore, we consider the computation of the 2–norm of vector $\mathbf{y}^T$ given by $||\mathbf{y}^T||_2 = \sqrt{\mathbf{y}\mathbf{y}^T}$. Fortunately, $||\mathbf{y}^T||_2$ can be obtained by computing inner products of two compact vectors in HTD format. Here, the only difference is that the two vectors are the same vector $\mathbf{y}^T$. The computation starts from the leaves of the binary tree and moves toward the root, requiring the same sequence of operations in the first part of the computation of reduced Gramians. But, this has already been discussed above.

Now, we can move to implementation issues regarding compact solution vectors in HTD format for Kronecker-based Markovian representations. The implementation [62] is within the `NSolve` package of the APNN toolbox [7, 22], relies on the basic functions of the `htucker` toolbox, and is available at [65]. The binary tree data structure accompanying the HTD format [62] is

allocated at the outset based on the value of H. It is stored in the form of an array of tree nodes from root to leaves level by level so that accessing the children of a parent node and vice versa becomes straightforward. In a tree node $\bar{t}$, there are pointers to matrices $U^{(\bar{t})}$ for leaves and $B^{(\bar{t})}$ for intermediate nodes which we have seen and accounted for before but also pointers to matrices $R^{(\bar{t})}$ and, as we explain shortly, (2×2) block matrices $M^{(\bar{t})}$ and $G^{(\bar{t})}$ for each node. Since we expect solution vectors to be dense, the matrices in the compact representation are stored as full matrices including those corresponding to the blocks of $M^{(\bar{t})}$ and $G^{(\bar{t})}$. The nonzero entries of the full matrices are kept in a one-dimensional floating-point array so that relevant `LAPACK` methods available at [248] can be called without having to copy vectors. We choose to store the transposes of matrices representing the compact solution vector in row sparse format (meaning they are stored by columns) so that relevant `LAPACK` methods can be called without having to transpose the input matrices.

The multiplication of the sparse Kronecker factors $A_k^{(h)}$ with the orthogonal basis matrices $U^{(h)}$ in $\mathbf{x}_{(k)}^T := \left(\bigotimes_{h=1}^{H} A_k^{(h)} U^{(h)}\right) \mathbf{c}$ is implemented using straightforward sparse matrix–vector multiplication. After the compact vector $\mathbf{x}_{(k)}^T$ is computed, the tree nodes of $\mathbf{y}_{(k)}^T$ are visited, and its respective fields are updated so that we have $\mathbf{y}_{(k+1)}^T$ at hand. Efficient computation of the reduced Gramian matrices $G^{(\bar{t})}$ for $\mathbf{y}_{(k+1)}^T$ as in [202] requires exploiting the block structure of the new transfer matrices $B^{(\bar{t})}$ whose blocks are already available in the corresponding tree nodes of $\mathbf{y}_{(k+1)}^T$ after the addition operation. Clearly, there is no need to generate block matrices (or cubic blocks as in [202]) with these blocks explicitly. We prefer to store $M^{(\bar{t})}$ and $G^{(\bar{t})}$ as (2×2) block matrices because of the add a term and then truncate approach followed. Let us next elaborate on this.

Assuming that $r_{\bar{t}}(\mathbf{y}_{(k)}^T)$ and $r_{\bar{t}}(\mathbf{x}_{(k)}^T)$ denote the ranks of matrices in compact representations of the two vectors that are summed up in node $\bar{t}$, $M^{(\bar{t})}$ and $G^{(\bar{t})}$ become $(r_{\bar{t}}(\mathbf{y}_{(k)}^T) r_{\bar{t}}(\mathbf{x}_{(k)}^T) \times r_{\bar{t}}(\mathbf{y}_{(k)}^T) r_{\bar{t}}(\mathbf{x}_{(k)}^T))$ matrices, where the first diagonal block is $(r_{\bar{t}}(\mathbf{y}_{(k)}^T) \times r_{\bar{t}}(\mathbf{y}_{(k)}^T))$ and the second diagonal block is $(r_{\bar{t}}(\mathbf{x}_{(k)}^T) \times r_{\bar{t}}(\mathbf{x}_{(k)}^T))$. Then the computation $M^{(\bar{t})} := (U^{(\bar{t})})^T U^{(\bar{t})}$ for leaf nodes can be formulated in (2×2) block manner as

$$M^{(\bar{t})}(s, s') := (U_s^{(\bar{t})})^T (U_{s'}^{(\bar{t})}) \quad \text{for } s, s' = 0, 1,$$

where $U_0^{(\bar{t})}$ and $U_1^{(\bar{t})}$ denote basis matrices of $\mathbf{y}_{(k)}$ and $\mathbf{x}_{(k)}$ at leaf node $\bar{t}$, respectively. This computation requires multiplying two full matrices for which the `dgemm` routine of `LAPACK` may be used. On the other hand, the computation $M^{(\bar{t})} := (B^{(\bar{t})})^T (M^{(\bar{t}_l)} \otimes M^{(\bar{t}_r)}) B^{(\bar{t})}$ for intermediate nodes can be formulated starting from the bottom of the tree to the root in (2×2) block manner as

$$M^{(\bar{t})}(s, s') := (B_s^{(\bar{t})})^T (M^{(\bar{t}_l)}(s, s') \otimes M^{(\bar{t}_r)}(s, s')) B_{s'}^{(\bar{t})} \quad \text{for } s, s' = 0, 1,$$

where $B_0^{(\bar{t})}$ and $B_1^{(\bar{t})}$ denote transfer matrices of $\mathbf{y}_{(k)}^T$ and $\mathbf{x}_{(k)}^T$ at node $\bar{t}$, respectively.

Similarly, we have reduced Gramian computations, but in opposite direction from root to leaves, that can be formulated in (2×2) block manner for $s, s' = 0, 1$ as $G^{(\bar{t})}(s, s') := 1$ when $\bar{t}$ corresponds to root; otherwise,

$$G^{(\bar{t}_l)}(s, s') := (B_s^{(\bar{t}:2,3)})^T (M^{(\bar{t}_r)}(s, s') \otimes G^{(\bar{t})}(s, s')) B_{s'}^{(\bar{t}:2,3)}$$

and

$$G^{(\bar{t}_r)}(s, s') := (B_s^{(\bar{t}:1,3)})^T (M^{(\bar{t}_l)}(s, s') \otimes G^{(\bar{t})}(s, s')) B_{s'}^{(\bar{t}:1,3)},$$

where $B_0^{(\bar{t}:2,3)}$, $B_0^{(\bar{t}:1,3)}$ are transfer matrices $B_0^{(\bar{t})}$ of $\mathbf{y}_{(k)}^T$ organized, respectively, as $(r_{\bar{t}_r}(\mathbf{y}_{(k)}^T) r_{\bar{t}}(\mathbf{y}_{(k)}^T) \times r_{\bar{t}_l}(\mathbf{y}_{(k)}^T))$, $(r_{\bar{t}_l}(\mathbf{y}_{(k)}^T) r_{\bar{t}}(\mathbf{y}_{(k)}^T) \times r_{\bar{t}_r}(\mathbf{y}_{(k)}^T)$ matrices, and $B_1^{(\bar{t}:2,3)}$, $B_1^{(\bar{t}:1,3)}$ are transfer matrices $B_1^{(\bar{t})}$ of $\mathbf{x}_{(k)}^T$ organized, respectively, as $(r_{\bar{t}_r}(\mathbf{x}_{(k)}^T) r_{\bar{t}}(\mathbf{x}_{(k)}^T) \times r_{\bar{t}_l}(\mathbf{x}_{(k)}^T))$, $(r_{\bar{t}_l}(\mathbf{x}_{(k)}^T) r_{\bar{t}}(\mathbf{x}_{(k)}^T) \times r_{\bar{t}_r}(\mathbf{x}_{(k)}^T))$ matrices. Such matrices are called *matricizations* of the given matrix (in this case, the transfer matrix $B_0^{(\bar{t})}$ or $B_1^{(\bar{t})}$ along specific dimensions) and, therefore, represent different organizations of the same data. We remark that the off-diagonal blocks of $M^{(\bar{t})}$ and $G^{(\bar{t})}$, respectively, satisfy the relationships $M^{(\bar{t})}(s, s') = (M^{(\bar{t})}(s', s))^T$ and $G^{(\bar{t})}(s, s') = (G^{(\bar{t})}(s', s))^T$. Therefore, only one off-diagonal block for these two matrices in each node needs to be computed. The computation of the three blocks of $M^{(\bar{t})}$ and $G^{(\bar{t})}$ requires multiplications using `dgemm` with matricizations and contraction of multidimensional data involving $B_s^{(\bar{t})}$ matrices for $s = 0, 1$ as discussed in [202]. We use two auxiliary vectors of length $\max_{\bar{t}_l, \bar{t}_r, \bar{t}}(r_{\bar{t}_l} r_{\bar{t}_r} r_{\bar{t}})$ to implement these operations. The disadvantage of not storing $M^{(\bar{t})}$ and $G^{(\bar{t})}$ as (2×2) block matrices is that longer auxiliary vectors would need to be allocated.

Truncation of a compact vector requires QR and singular value decompositions [158, 302, 303] to be performed [202]. In order to compute these decompositions, `dgeqrf` and `dgesdd` routines of `LAPACK` are used. Since we expect input matrices to be dense, we do not call routines expecting sparse matrices. For a leaf node $\bar{t}$, the $(n_{\bar{t}} \times (r_{\bar{t}}(\mathbf{y}_{(k)}^T) + r_{\bar{t}}(\mathbf{x}_{(k)}^T)))$ input matrix $U^{(\bar{t})}$ may be obtained by concatenating the matrices $U_0^{(\bar{t})}$ and $U_1^{(\bar{t})}$ corresponding to $\mathbf{y}_{(k)}^T$ and $\mathbf{x}_{(k)}^T$, respectively. Since the input matrix is also an output matrix, the upper-triangular factor $R^{(\bar{t})}$ of the QR decomposition is returned from `dgeqrf` in the upper-triangular part of the input matrix in which the lower-triangular part has the Householder reflections amounting to the orthogonal factor $Q^{(\bar{t})}$ (not to be confused with the generator matrix and any related matrix thereof). After $R^{(\bar{t})}$ is obtained, $R^{(\bar{t})} G^{(\bar{t})} (R^{(\bar{t})})^T$ needs to be formed. To this end, we first transform the block matrix $G^{(\bar{t})}$ to a dense matrix (with a single block) and multiply this new matrix held as a one-dimensional array from left and right using the `dtrmm` routine of `LAPACK`. Note that `dtrmm` does not accept a trapezoid $R^{(\bar{t})}$; however, this case can be handled by multiplying

triangular and rectangular parts of $R^{(\bar{t})}$ separately using `dtrmm` and `dgemm`. Hence, there is no need to copy the output of `dgeqrf` to another matrix including $R^{(\bar{t})}$. Once $R^{(\bar{t})}G^{(\bar{t})}(R^{(\bar{t})})^T$ is formed, it needs to be decomposed for its singular values and vectors. To do this, we prefer to use the `dgesdd` routine over the `dgesvd` routine since it is said to be faster [248]. We remark that this routine computes singular values through the *symmetric eigenvalue decomposition* [125, 158, 303] and the singular vectors are truncated at a certain number or possibly by omitting some corresponding to the smaller singular values based on an error tolerance. $S^{(\bar{t})}$ ends up being the matrix holding the $r_{\bar{t}}$ singular vectors. Then the orthogonal basis matrix $U^{(\bar{t})} = Q^{(\bar{t})}S^{(\bar{t})}$ is computed using the `dormqr` routine. In order to avoid storing $S^{(\bar{t})}$, we prefer to update $R^{(\bar{t})}$ with $(S^{(\bar{t})})^T$ as in the `htucker` toolbox [203].

The same sequence of operations are carried out level by level from the parents of the leaves to the top of the tree excluding the root. The product $(S^{(\bar{t})})^T R^{(\bar{t})}$ is computed using `dtrmm`, possibly with an additional call to `dgemm` when $R^{(\bar{t})}$ is trapezoid, and stored in the matrix that was allocated for $R^{(\bar{t})}$. Note that $(S^{(\bar{t})})^T R^{(\bar{t})} = (F_0^{(\bar{t})}\ F_1^{(\bar{t})})$ is an $(r_{\bar{t}} \times (r_{\bar{t}}(\mathbf{y}_{(k)}^T) + r_{\bar{t}}(\mathbf{x}_{(k)}^T)))$ matrix with the two blocks $F_s^{(\bar{t})}$ for $s = 0, 1$, where $r_{\bar{t}}$ is the rank of node $\bar{t}$ after truncation. Then for a non-leaf node $\bar{t}$, the QR factorization of $\sum_{s=0}^{1}(F_s^{(\bar{t}_l)} \otimes F_s^{(\bar{t}_r)})B_s^{(\bar{t})}$ needs to be computed. This computation requires multiplications using `dgemm` with matricizations of multidimensional data involving $B_s^{(\bar{t})}$ matrices for $s = 0, 1$ as discussed in [202]. Finally, the transfer matrix $B^{(\bar{t})} = Q^{(\bar{t})}S^{(\bar{t})}$ is computed using `dormqr`.

In [64], numerical experiments are performed on the availability and polling models in Chapter 2 to observe how the memory requirements of the compact solution vector in HTD format change over the course of iterations due to the sequence of multiply, add, and truncate operations in each iteration, together with the average time it takes to perform the iteration and the influence of the truncation error on the quality of the solution. To facilitate this, the compact solution vector in HTD format is iteratively multiplied with the uniformized generator matrix of the respective CTMC in Kronecker form until a predetermined stopping criterion is met starting from an initial solution. The same numerical experiment is performed with a full solution vector of length equal to the reachable state space size, $|\mathcal{R}|$, using the modified shuffle algorithm. The two approaches are compared for their memory and timing requirements, leading us to the conclusion that compact vectors in HTD format become relatively more memory efficient as the number of dimensions increases.

In particular, we consider the following iteration steps of the *power method* [17, 192, 305]

$$\boldsymbol{\pi}_{(it+1)} := \boldsymbol{\pi}_{(it)}P \quad \text{for } it = 0, \ldots, \texttt{maxit} - 1$$

starting with the uniform distribution in $\boldsymbol{\pi}_{(0)}$, where

$$P := I + \Delta Q, \quad \Delta := 0.999/\max_{i \in \mathcal{R}} |q_{i,i}|, \tag{5.3}$$

and `maxit` is the maximum number of iterations to be performed. The scalar 0.999 ensures a positive diagonal in P. The convergence rate of this method is known to depend on the magnitude of the subdominant eigenvalue of P [305]. Therefore, it is not the most efficient solution method for steady-state analysis, a subject to which we return in Chapter 6 in the context of point iterative methods. But, similar iteration steps can be applied in more advanced iterative methods, and they can be directly used in *uniformization* [164, 169, 275], which is practically the simplest method to implement for transient analysis [305]. We discuss this method and others in Chapter 7.

At iteration it where we compute the solution vector $\boldsymbol{\pi}_{(it+1)}$, the *residual vector* and the *error vector*, respectively, satisfy

$$\mathbf{r}_{(it+1)} = -\boldsymbol{\pi}_{(it+1)}Q \quad \text{and} \quad \mathbf{e}_{(it+1)} = \boldsymbol{\pi}_{(it+1)} - \boldsymbol{\pi}_{(it)}. \tag{5.4}$$

Note that it is possible to rewrite the error vector as $\mathbf{e}_{(it+1)} = -\Delta\mathbf{r}_{(it)}$, the *negated scaled residual vector* corresponding to the previous solution vector. Hence, the 2–norm of the error at iteration it can be obtained after computing the negated scaled residual at the previous iteration.

The parameters of the availability and polling models considered in the experiments are given as follows. In the availability model in Figure 2.2 considered here, components are repaired by a global repair facility with preemptive priority such that components from submodel 1 have the highest priority and components from submodel H have the least priority, but there is no priority for repair among different components in the same submodel. The rates are $\lambda_1^{(h)} := 5 \times 10^{-4}$, $\lambda_2^{(h)} := 4 \times 10^{-4}$, and $\lambda_3^{(h)} := 10^{-4}$ for $h = 1, \ldots, H$, $\mu_1^{(1)} := 1$, $\mu_2^{(1)} := 2$, $\mu_3^{(1)} := 4$, $\mu_1^{(h)} := 0.1$, $\mu_2^{(h)} := 0.2$, and $\mu_3^{(h)} := 0.4$ for $h = 2, \ldots, H$. The polling model in Figure 2.5 considered here has state-independent transition rates; there can be at most one customer being served at a time in each queue and at most one server traveling at a time from each queue to the next queue. The parameters are $S := 2$, $C_h := 10$, $\mu_h := 1$, $\gamma_h := 10$ for $h = 1, \ldots, H$, and $\sum_{h=1}^{H} \lambda_h := 1.5$.

The 2–norm of the final error vector turns out to be the same for the full and HTD representations. Due to the reduced memory requirements of the HTD solution vector for larger values of H, the compact representation results in smaller iteration times when H increases. Ranks of the different matrices forming the HTD of the solution vector in the availability model with different values of H are inspected for `trunc_tol` $:= 10^{-8}$ and `maxit` := 1,000. It is seen that the ranks of the matrices in HTD format remain moderate across iterations. The memory allocated to the matrices and the workspace to carry out the HTD computation is fairly small. The matrices stored in the HTD format are quite dense such that their sparse storage is not required.

Chapter 6
Steady-State Analysis

Classical *iterative methods* for the solution of a linear system of equations as in (1.3) start with an initial approximation. At each iteration, they modify the entries of the current approximation in a particular way to obtain a new approximation with the objective that the approximations eventually converge to the true solution [168, 280]. These methods are the building blocks of all advanced iterative methods and can be expressed through the multiplication of the current solution vector at a given iteration with a particular matrix that can be obtained at the outset by splitting the coefficient matrix of the linear system [305, 322], which is Q in our setting. Therefore, we begin by splitting the smaller matrices that form the Kronecker products as in [315] and show how classical iterative methods can be formulated in terms of these smaller matrices. We present block versions of the methods since point versions follow from the block versions by considering blocks of order one.

We continue the discussion with *projection* (or *Krylov subspace*) methods for MCs based on Kronecker products in which approximate solutions satisfying various constraints are extracted from small dimensional subspaces [17, 278, 305]. Being iterative, their basic operation is also vector–Kronecker product multiplication. However, compared to block iterative methods, they require a larger number of supplementary vectors of length equal to the reachable state space size. But, more importantly they need to be used with *preconditioners* to result in effective solvers. Fortunately, the first term of the splitting associated with block iterative methods can be used as preconditioner [60].

In [43, 49, 50, 52], *aggregation–disaggregation* steps are coupled with various iterative methods for MCs based on Kronecker products to accelerate convergence. An *iterative aggregation–disaggregation* (IAD) method for MCs based on Kronecker products and its adaptive version, which analyzes aggregated systems for those parts where the error is estimated to be high, are proposed in [47] and [48], respectively. The adaptive IAD method in [48] is

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_6

improved in [53] through a recursive definition and called *multilevel* (ML). Here, we present this simple ML method and then discuss a class of ML methods based on it which are shown to be quite effective [59, 61] in solving a large number of problems in the literature. ML methods can easily use iterative methods based on splittings at each level before aggregation and after disaggregation.

Matrix analytic methods are geared toward MCs having state spaces that can be partitioned into subsets called *levels*. For such MCs, the transition matrix when symmetrically permuted according to increasing level number should also have a particular nonzero structure, such as *block tridiagonal* or *block Hessenberg*. For instance, the well-known *quasi-birth-and-death processes* (QBDs) fall under the class of processes which lend themselves to steady-state analysis with matrix analytic methods. In this context, matrix analytic methods are those for which equations involving matrices are set up and solutions are expressed in terms of matrices [166]. These methods were originally proposed [251, 252] for processes with PH distributions and then improved over the years [32, 166, 214]. They characterize the solution by matrices having stochastic interpretations and sizes determined by the number of states within levels. Here, we consider CTMCs and concentrate on the class of *level-dependent* QBDs (LDQBDs). We show how the systems of stochastic chemical kinetics modeled using Kronecker products can be expressed as infinite LDQBDs, and analyzed for their steady-state with the help of Lyapunov functions. In passing, we remark that the concept of level introduced here has nothing to do with the level concept introduced during the discussion of iterative solution methods.

Decompositonal iterative methods compute steady-state solutions by decomposing a model into its submodels, analyzing the submodels individually for their steady-state, and putting back the individual solutions together in an iterative computational framework. As such, they may aim at obtaining the steady-state solution exactly up to computer precision, approximatively when a few digits of accuracy is sufficient, or within upper and lower bounds. Methods of the first two kinds are discussed on the simple availability model in Example 1 [15] and a class of closed queueing networks [104].

Finally, we describe how compact solution vectors in HTD format can be used in the analysis of point iterative methods and projection methods following the recent results in [64, 66].

6.1 Block Iterative Methods

We begin by splitting submodel transition matrices into three terms as in

$$Q_k^{(h)} = D_k^{(h)} + U_k^{(h)} + L_k^{(h)} \quad \text{for } k \in \mathcal{K} \text{ and } h = 1, \ldots, H \,, \tag{6.1}$$

where $D_k^{(h)}$, $U_k^{(h)}$, and $L_k^{(h)}$ are respectively the diagonal, strictly upper-triangular, and strictly lower-triangular parts of $Q_k^{(h)}$. Observe that $D_k^{(h)} \geqslant 0$, $U_k^{(h)} \geqslant 0$, and $L_k^{(h)} \geqslant 0$ since $Q_k^{(h)} \geqslant 0$.

Now, let us denote the off-diagonal part of Q in (2.10) as

$$Q_O = Q - Q_D \quad \text{so that} \quad Q_O(p,w) = \sum_{k \in \mathcal{K}_{p,w}} Q_k(p,w), \tag{6.2}$$

where

$$Q_k(p,w) = \alpha_k \bigotimes_{h=1}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) \quad \text{for } p,w = 0,\ldots,N-1\,.$$

Observe that $Q_O(p,w)$ for $p > w$ is in the strictly block lower-triangular part of Q_O, while $Q_O(p,w)$ for $p < w$ is in the strictly block upper-triangular part of Q_O. Hence, all that needs to be done is to provide block splittings of $Q_O(p,p)$ for $p = 0,\ldots,N-1$.

Using Lemma A.8 in [315], which rests on the associativity of Kronecker product and the *distributivity* of Kronecker product over matrix addition, it is possible to express diagonal block $Q_O(p,p)$ in Q_O at level $l = 0,\ldots,H$ using (6.1) as

$$Q_O(p,p) = Q(p,p)_{U(l)} + Q(p,p)_{L(l)} + Q(p,p)_{DU(l)} + Q(p,p)_{DL(l)}\,, \tag{6.3}$$

where

$$\begin{aligned} Q(p,p)_{U(l)} = \sum_{k \in \mathcal{K}_{p,p}} \sum_{h=1}^{l} \alpha_k \left(\bigotimes_{h'=1}^{h-1} D_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right) \otimes U_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) \\ \otimes \left(\bigotimes_{h'=h+1}^{H} Q_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right)\,, \end{aligned} \tag{6.4}$$

$$\begin{aligned} Q(p,p)_{L(l)} = \sum_{k \in \mathcal{K}_{p,p}} \sum_{h=1}^{l} \alpha_k \left(\bigotimes_{h'=1}^{h-1} D_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right) \otimes L_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) \\ \otimes \left(\bigotimes_{h'=h+1}^{H} Q_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right) \end{aligned} \tag{6.5}$$

correspond, respectively, to the strictly block upper- and block lower-triangular parts of $Q_O(p,p)$ at level l, and

$$\begin{aligned} Q(p,p)_{DU(l)} = \sum_{k \in \mathcal{K}_{p,p}} \sum_{h=l+1}^{H} \alpha_k \left(\bigotimes_{h'=1}^{h-1} D_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right) \otimes U_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) \\ \otimes \left(\bigotimes_{h'=h+1}^{H} Q_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right)\,, \end{aligned} \tag{6.6}$$

$$Q(p,p)_{DL(l)} = \sum_{k\in\mathcal{K}_{p,p}} \sum_{h=l+1}^{H} \alpha_k \left(\bigotimes_{h'=1}^{h-1} D_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right) \otimes L_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) \otimes \left(\bigotimes_{h'=h+1}^{H} Q_k^{(h')}(\mathcal{R}_p^{(h')}, \mathcal{R}_p^{(h')}) \right) . \tag{6.7}$$

correspond, respectively, to the strictly upper- and strictly lower-triangular parts of the *block diagonal* of $Q_O(p,p)$ at level l. Observe that $Q(p,p)_{U(l)} \geqslant 0$, $Q(p,p)_{L(l)} \geqslant 0$, $Q(p,p)_{DU(l)} \geqslant 0$, and $Q(p,p)_{DL(l)} \geqslant 0$. Furthermore, we remark that $l = 0$ implies $Q_O(p,p)$ is a single block for which $Q(p,p)_{U(0)} = Q(p,p)_{L(0)} = 0$, whereas $l = H$ corresponds to a point-wise partitioning of $Q_O(p,p)$ for which $Q(p,p)_{DU(H)} = Q(p,p)_{DL(H)} = 0$. Hence, for iterative methods based on *block partitionings* $l = 1, \ldots, H-1$ should be used. In passing to an example, we remark that in a non-Kronecker setting, one could obtain block partitionings of Q as discussed in [78, 123, 253]. Here, it is the Kronecker structure of Q that suggests suitable block partitionings.

Example 2. (ctnd.) Consider the block partitioning of our problem at level 0 for which $l = 0$ and Q_O is viewed as a (4×4) block matrix with four blocks, respectively, of order 4, 2, 4, and 2 along the diagonal. Then for $p = 0, 1, 2, 3$ from (6.4) and (6.5), we have

$$Q(p,p)_{U(0)} = 0 \,, \quad Q(p,p)_{L(0)} = 0 \,, \quad Q(p,p)_{U(0)} + Q(p,p)_{L(0)} = 0 \,,$$

whereas from (6.6) and (6.7), we have

$$\begin{aligned} Q(p,p)_{DU(0)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k \Big(& U_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_p^{(1)}) \otimes Q_k^{(2)}(\mathcal{R}_p^{(2)}, \mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)}, \mathcal{R}_p^{(3)}) \\ & + D_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_p^{(1)}) \otimes U_k^{(2)}(\mathcal{R}_p^{(2)}, \mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)}, \mathcal{R}_p^{(3)}) \\ & + D_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_p^{(1)}) \otimes D_k^{(2)}(\mathcal{R}_p^{(2)}, \mathcal{R}_p^{(2)}) \otimes U_k^{(3)}(\mathcal{R}_p^{(3)}, \mathcal{R}_p^{(3)}) \Big) \,, \\ Q(p,p)_{DL(0)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k \Big(& L_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_p^{(1)}) \otimes Q_k^{(2)}(\mathcal{R}_p^{(2)}, \mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)}, \mathcal{R}_p^{(3)}) \\ & + D_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_p^{(1)}) \otimes L_k^{(2)}(\mathcal{R}_p^{(2)}, \mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)}, \mathcal{R}_p^{(3)}) \\ & + D_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_p^{(1)}) \otimes D_k^{(2)}(\mathcal{R}_p^{(2)}, \mathcal{R}_p^{(2)}) \otimes L_k^{(3)}(\mathcal{R}_p^{(3)}, \mathcal{R}_p^{(3)}) \Big) \,, \end{aligned}$$

$$Q(p,p)_{DU(0)} + Q(p,p)_{DL(0)} = Q_O(p,p) \,.$$

Now consider the block partitioning of the problem at level 1 for which $l = 1$ and $Q_O(0,0)$ is viewed as a (2×2) block matrix with blocks of order 2, $Q_O(1,1)$ is viewed as a (1×1) block matrix with a block of order 2, $Q_O(2,2)$ is viewed as a (2×2) block matrix with blocks of order 2, and $Q_O(3,3)$ is viewed as a (2×2) block matrix with blocks of order 1 (see Table 2.1). Then from (6.4) and (6.5), we have

$$Q(p,p)_{U(1)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k U_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes Q_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \,,$$
$$Q(p,p)_{L(1)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k L_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes Q_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \,.$$

Consequently,

$$Q(0,0)_{U(1)} + Q(0,0)_{L(1)} = \begin{pmatrix} & \lambda_1 \\ & \lambda_1 \\ \mu_1 & \\ \mu_1 & \end{pmatrix}, \quad Q(1,1)_{U(1)} + Q(1,1)_{L(1)} = 0,$$

$$Q(2,2)_{U(1)} + Q(2,2)_{L(1)} = \begin{pmatrix} & & \\ \mu_1 & & \\ & \mu_1 & \end{pmatrix}, \quad \text{and}$$
$$Q(3,3)_{U(1)} + Q(3,3)_{L(1)} = \begin{pmatrix} & \\ \mu_1 & \end{pmatrix} \,,$$

whereas from (6.6) and (6.7), we have

$$\begin{aligned} Q(p,p)_{DU(1)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k \Big(& D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes U_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \\ & + D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes D_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes U_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \Big) \,, \\ Q(p,p)_{DL(1)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k \Big(& D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes L_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \\ & + D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes D_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes L_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \Big) \,. \end{aligned}$$

Consequently,

$$Q(0,0)_{DU(1)} + Q(0,0)_{DL(1)} = \begin{pmatrix} & 2\lambda_3 & \\ \mu_3 & & \\ & & 2\lambda_3 \\ & \mu_3 & \end{pmatrix},$$
$$Q(1,1)_{DU(1)} + Q(1,1)_{DL(1)} = \begin{pmatrix} & \\ \mu_3 & \end{pmatrix} \,,$$
$$Q(2,2)_{DU(1)} + Q(2,2)_{DL(1)} = \begin{pmatrix} \mu_3 & & \\ & & \\ & \mu_3 & \end{pmatrix}, \quad \text{and}$$
$$Q(3,3)_{DU(1)} + Q(3,3)_{DL(1)} = 0 \,.$$

Finally, consider the block partitioning of the same example at level 2 for which $l = 2$. Then from (6.4) and (6.5), we have

$$\begin{aligned}
Q(p,p)_{U(2)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k \Big(& U_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes Q_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \\
& + D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes U_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \Big), \\
Q(p,p)_{L(2)} = \sum_{k\in\mathcal{K}_{p,p}} \alpha_k \Big(& L_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes Q_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \\
& + D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes L_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes Q_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}) \Big),
\end{aligned}$$

whereas from (6.6) and (6.7), we have

$$\begin{aligned}
Q(p,p)_{DU(2)} &= \sum_{k\in\mathcal{K}_{p,p}} \alpha_k D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes D_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes U_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}), \\
Q(p,p)_{DL(2)} &= \sum_{k\in\mathcal{K}_{p,p}} \alpha_k D_k^{(1)}(\mathcal{R}_p^{(1)},\mathcal{R}_p^{(1)}) \otimes D_k^{(2)}(\mathcal{R}_p^{(2)},\mathcal{R}_p^{(2)}) \otimes L_k^{(3)}(\mathcal{R}_p^{(3)},\mathcal{R}_p^{(3)}).
\end{aligned}$$

Note that at this level, as in $l = 1$, for this particular example $Q_O(0,0)$ is viewed as a (2×2) block matrix with blocks of order 2, $Q_O(1,1)$ is viewed as a (1×1) block matrix with a block of order 2, $Q_O(2,2)$ is viewed as a (2×2) block matrix with blocks of order 2, and $Q_O(3,3)$ is viewed as a (2×2) block matrix with blocks of order 1 (see Table 2.1). Hence,

$$\begin{aligned}
Q(0,0)_{U(2)} + Q(0,0)_{L(2)} &= Q(0,0)_{U(1)} + Q(0,0)_{L(1)}\ , \\
Q(1,1)_{U(2)} + Q(1,1)_{L(2)} &= Q(1,1)_{U(1)} + Q(1,1)_{L(1)} = 0, \\
Q(2,2)_{U(2)} + Q(2,2)_{L(2)} &= Q(2,2)_{U(1)} + Q(2,2)_{L(1)}\ , \\
Q(3,3)_{U(2)} + Q(3,3)_{L(2)} &= Q(3,3)_{U(1)} + Q(3,3)_{L(1)}\ , \\
Q(0,0)_{DU(2)} + Q(0,0)_{DL(2)} &= Q(0,0)_{DU(1)} + Q(0,0)_{DL(1)}\ , \\
Q(1,1)_{DU(2)} + Q(1,1)_{DL(2)} &= Q(1,1)_{DU(1)} + Q(1,1)_{DL(1)}\ , \\
Q(2,2)_{DU(2)} + Q(2,2)_{DL(2)} &= Q(2,2)_{DU(1)} + Q(2,2)_{DL(1)}\ , \\
Q(3,3)_{DU(2)} + Q(3,3)_{DL(2)} &= Q(3,3)_{DU(1)} + Q(3,3)_{DL(1)} = 0.
\end{aligned}$$

At $l = 3$, diagonal blocks $Q_O(0,0)$, $Q_O(1,1)$, $Q_O(2,2)$, and $Q_O(3,3)$ of Q_O are, respectively, (4×4), (2×2), (4×4), and (2×2) block matrices with blocks of order 1.

Now, let Q in (2.10) be irreducible and *split* at level l using (6.3) as

$$Q = Q_O + Q_D = Q_{U(l)} + Q_{L(l)} + Q_{DU(l)} + Q_{DL(l)} + Q_D = M - W\ , \quad (6.8)$$

where

$$Q_{U(l)} = \begin{pmatrix} Q(0,0)_{U(l)} & Q(0,1) & \cdots & Q(0,N-1) \\ & Q(1,1)_{U(l)} & \cdots & Q(1,N-1) \\ & & \ddots & \vdots \\ & & & Q(N-1,N-1)_{U(l)} \end{pmatrix}$$

is the strictly block upper-triangular part of Q_O at level l,

$$Q_{L(l)} = \begin{pmatrix} Q(0,0)_{L(l)} & & & \\ \vdots & \ddots & & \\ Q(N-2,0) & \cdots & Q(N-2,N-2)_{L(l)} & \\ Q(N-1,0) & \cdots & Q(N-1,N-2) & Q(N-1,N-1)_{L(l)} \end{pmatrix}$$

is the strictly block lower-triangular part of Q_O at level l,

$$Q_{DU(l)} = \begin{pmatrix} Q(0,0)_{DU(l)} & & \\ & \ddots & \\ & & Q(N-1,N-1)_{DU(l)} \end{pmatrix}$$

is the strictly upper-triangular part of the block diagonal of Q_O at level l,

$$Q_{DL(l)} = \begin{pmatrix} Q(0,0)_{DL(l)} & & \\ & \ddots & \\ & & Q(N-1,N-1)_{DL(l)} \end{pmatrix}$$

is the strictly lower-triangular part of the block diagonal of Q_O at level l, and M is nonsingular (i.e., M^{-1} exists).

Then *power*, *block Jacobi over-relaxation* (BJOR), and *block successive over-relaxation* (BSOR) methods are based on different splittings of Q [322], and each method is in the form

$$\boldsymbol{\pi}_{(it+1)} := \boldsymbol{\pi}_{(it)} T \quad \text{for} \quad it = 0, \ldots, \texttt{maxit} - 1$$

with the sequence of approximations $\boldsymbol{\pi}_{(it+1)}$ to the steady-state vector $\boldsymbol{\pi}$ in (1.3), where $\boldsymbol{\pi}_{(0)} \in \mathbb{R}_{>0}^{1\times|\mathcal{R}|}$ is the initial approximation such that $\boldsymbol{\pi}_{(0)}\mathbf{e} = 1$ and

$$T = WM^{-1}$$

is the *iteration matrix*.

Note that T does not change from iteration to iteration, and only the current approximation $\boldsymbol{\pi}_{(it)}$ is used to compute the new approximation $\boldsymbol{\pi}_{(it+1)}$. Hence, these methods based on splittings of the coefficient matrix are also known as *stationary* iterative methods. Since Q is a singular matrix and assumed to be irreducible, the largest eigenvalue [158, 242] of T in magnitude is 1, that is, the *spectral radius* of T, $\rho(T)$, is equal to 1. In order to ensure convergence, T should not have other eigenvalues with magnitude 1, that is,

T should be *aperiodic*. For a converging iteration, the magnitude of the eigenvalue of T closest to 1, that is, its *subdominant eigenvalue*, determines the *rate of convergence* [17, 30, 229, 305, 322]. When the subdominant eigenvalue of T is close to 1 in magnitude, slow convergence is witnessed.

Given a DTMC with one-step transition probability matrix P such that $P \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}|\times|\mathcal{R}|}$ and $P\mathbf{e} = \mathbf{e}$ [194, 305], one can conceive of block iterative methods for $Q = P - I$ as those defined by (6.8) to compute the steady-state vector $\boldsymbol{\pi}$ which satisfies

$$\boldsymbol{\pi}P = \boldsymbol{\pi}, \quad \boldsymbol{\pi}\mathbf{e} = 1. \tag{6.9}$$

In practice, an explicit inversion of M does not take place. Instead, at iteration it, one solves the consistent linear system

$$\boldsymbol{\pi}_{(it+1)}M = \boldsymbol{\pi}_{(it)}W$$

with coefficient matrix M for the unknown vector $\boldsymbol{\pi}_{(it+1)}$ using the right-hand side vector $\boldsymbol{\pi}_{(it)}W$. The iteration stops if the norm of the error vector (or alternatively, the norm of the residual vector) (see (5.4)) is less than a prespecified tolerance, `stop_tol`, a run time limit, `time_limit`, is reached, or a maximum number of iterations, `maxit`, is performed. Otherwise, the iteration number it is incremented and the iteration continues.

The particular splittings corresponding to power, BJOR, and (*forward*) BSOR methods are

$$\begin{aligned}
M^{Power} &= -\Gamma I\,, \\
W^{Power} &= -\Gamma\left(I + \frac{1}{\Gamma}Q\right)\,, \\
M^{BJOR} &= \frac{1}{\omega}(Q_D + Q_{DU(l)} + Q_{DL(l)})\,, \\
W^{BJOR} &= \frac{1-\omega}{\omega}(Q_D + Q_{DU(l)} + Q_{DL(l)}) - Q_{U(l)} - Q_{L(l)}\,, \\
M^{BSOR} &= \frac{1}{\omega}(Q_D + Q_{DU(l)} + Q_{DL(l)}) + Q_{U(l)}\,, \\
W^{BSOR} &= \frac{1-\omega}{\omega}(Q_D + Q_{DU(l)} + Q_{DL(l)}) - Q_{L(l)}\,,
\end{aligned}$$

where $\Gamma \in [\max_{i\in\mathcal{R}} |q_D(i,i)|, \infty)$ is the *uniformization parameter* of power method and $\omega \in (0,2)$ is the *relaxation parameter* of BJOR and BSOR methods. Here, forward iteration refers to computing unknowns ordered toward the beginning of the reachable state space earlier than unknowns ordered later in the reachable state space. Power method works at level $l = H$ since it is a point method. Furthermore, BJOR and BSOR reduce to *block Jacobi* (BJacobi) and *block Gauss–Seidel* (BGS) methods for $\omega = 1$, and they become point JOR and point SOR methods for $l = H$. We remark that [173] shows how one can find $\max_{i\in\mathcal{R}} |q_D(i,i)|$ in the presence of functional transitions when Q_D is given as a sum of Kronecker products. It is possible to use the same approach in each $Q_D(p,p)$ for $p = 0,\ldots,N-1$ when there are multiple reachable state space partitions.

When Q is irreducible and $\omega \in (0,1)$, for JOR and SOR we have $M^{-1}, W \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}|\times|\mathcal{R}|}$, $\rho(T) = 1$, T is irreducible and aperiodic. Hence, JOR and SOR can be made to converge by choosing $\omega \in (0,1)$. To avoid periodicity of T^{GS} for the point GS method, one can symmetrically permute Q such that it is in block lower-Hessenberg form (i.e., $Q(p,w) = 0$ if $p+1 < w$) and all subdiagonal blocks have at least one nonzero in each row with the last diagonal block being (1×1) [96]. When Q is irreducible, $T^{BJacobi}$ and T^{BGS} satisfy $M^{-1}, W \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}|\times|\mathcal{R}|}$ and $\rho(T) = 1$ is a simple eigenvalue [16, 230, 232, 233, 249].

Now, let us assume that $T \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}|\times|\mathcal{R}|}$ is irreducible, but periodic having $N' \in \mathbb{Z}_{>0}$ *periodic classes*, without loss of generality, as in

$$T = \begin{pmatrix} & T(0,1) & & \\ & & \ddots & \\ & & & T(N'-2,N'-1) \\ T(N'-1,0) & & & \end{pmatrix}.$$

Note that $N' = 1$ is the aperiodic case. BJacobi and BGS may yield a periodic T with $N' > 1$ when

$$Q = \begin{pmatrix} Q(0,0) & Q(0,1) & & \\ & \ddots & \ddots & \\ & & Q(N'-2,N'-2) & Q(N'-2,N'-1) \\ Q(N'-1,0) & & & Q(N'-1,N'-1) \end{pmatrix}.$$

An often overlooked result in this context is that when none of the diagonal blocks of Q can be symmetrically permuted to block diagonal form, $T^{BJacobi}$ and T^{BGS} will have states of each partition in the same periodic class [87]. Researchers have looked into ways of avoiding periodicity of T and accelerating convergence. This is something to which we return in the next sections.

Since $Q = Q_O + Q_D$, power method at iteration it can be expressed as

$$\boldsymbol{\pi}_{(it+1)} := \boldsymbol{\pi}_{(it)} + \frac{1}{\Gamma}\left(\boldsymbol{\pi}_{(it)}Q_D + \boldsymbol{\pi}_{(it)}Q_O\right) . \tag{6.10}$$

Observe that the second term in (6.10) poses no problem from a computational point of view since Q_D is diagonal, and the third term can be efficiently implemented using the vector–Kronecker product multiplication algorithm since Q_O is expressed using sums of Kronecker products (see (6.2)).

The BJOR method with a level l block partitioning at iteration it satisfies

$$\begin{aligned} &\boldsymbol{\pi}_{(it+1)}(Q_D + Q_{DU(l)} + Q_{DL(l)}) \\ &\quad = (1-\omega)\left(\boldsymbol{\pi}_{(it)}Q_D + \boldsymbol{\pi}_{(it)}Q_{DU(l)} + \boldsymbol{\pi}_{(it)}Q_{DL(l)}\right) \\ &\qquad -\omega\left(\boldsymbol{\pi}_{(it)}Q_{U(l)} + \boldsymbol{\pi}_{(it)}Q_{L(l)}\right) . \end{aligned} \tag{6.11}$$

This is a block diagonal linear system with nonsingular coefficient matrix $(Q_D + Q_{DU(l)} + Q_{DL(l)})$ and a nonzero right-hand side which can be efficiently computed using the vector–Kronecker product multiplication algorithm, since $Q_{U(l)}$, $Q_{L(l)}$, $Q_{DU(l)}$, and $Q_{DL(l)}$ are expressed using sums of Kronecker products (see (6.4), (6.5), (6.6), and (6.7)). In particular, there are $\prod_{h=1}^{l} |\mathcal{R}_p^{(h)}|$ blocks of order $\prod_{h=l+1}^{H} |\mathcal{R}_p^{(h)}|$ along the diagonal of $Q_D(p,p) + Q(p,p)_{DU(l)} + Q(p,p)_{DL(l)}$ for $p = 0, \ldots, N-1$. Hence, (6.11) is equivalent to $\sum_{p=0}^{N-1} \prod_{h=1}^{l} |\mathcal{R}_p^{(h)}|$ independent, nonsingular linear systems with nonzero right-hand sides where the linear systems with coefficient matrices in $Q_D(p,p) + Q(p,p)_{DU(l)} + Q(p,p)_{DL(l)}$ are each of order $\prod_{h=l+1}^{H} |\mathcal{R}_p^{(h)}|$. If there is space, one can generate and factorize in sparse storage the nonsingular blocks of the form

$$Q((i_1,\ldots,i_l),(i_1,\ldots,i_l)) = \sum_{k \in \mathcal{K}_{p,p}} \alpha_k \left(\prod_{h=1}^{l} q_k^{(h)}(i_h,i_h) \right) \left(\bigotimes_{h=l+1}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)},\mathcal{R}_p^{(h)}) \right)$$

$$+ \; Q_D((i_1,\ldots,i_l),(i_1,\ldots,i_l)) \quad \text{for } (i_1,\ldots,i_l) \in \bigtimes_{h=1}^{l} \mathcal{R}_p^{(h)} \text{ and } p = 0,\ldots,N-1 \tag{6.12}$$

along the diagonal (see (2.4)) of $(Q_D + Q_{DU(l)} + Q_{DL(l)})$ at the outset and solve the $\sum_{p=0}^{N-1} |\bigtimes_{h=1}^{l} \mathcal{R}_p^{(h)}|$ systems directly at each iteration. Otherwise, one can use an iterative method, even a block iterative method, such as BJOR, since the off-diagonal parts of diagonal blocks given by

$$\sum_{k \in \mathcal{K}_{p,p}} \alpha_k \left(\prod_{h=1}^{l} q_k^{(h)}(i_h,i_h) \right) \left(\bigotimes_{h=l+1}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)},\mathcal{R}_p^{(h)}) \right)$$

are sums of Kronecker products.

The situation with the BSOR method is not very different from that of BJOR. For BSOR with a level l block partitioning, at iteration it, we have

$$\boldsymbol{\pi}_{(it+1)}(Q_D + Q_{DU(l)} + Q_{DL(l)} + \omega Q_{U(l)}) \tag{6.13}$$

$$= (1-\omega)\left(\boldsymbol{\pi}_{(it)} Q_D + \boldsymbol{\pi}_{(it)} Q_{DU(l)} + \boldsymbol{\pi}_{(it)} Q_{DL(l)}\right) - \omega \boldsymbol{\pi}_{(it)} Q_{L(l)} \; .$$

This is a block upper-triangular linear system with the nonsingular coefficient matrix $(Q_D + Q_{DU(l)} + Q_{DL(l)} + \omega Q_{U(l)})$ and a nonzero right-hand side which can be efficiently computed using the vector–Kronecker product multiplication algorithm, since $Q_{L(l)}$, $Q_{DU(l)}$, and $Q_{DL(l)}$ are expressed using sums of Kronecker products. In particular, there are $\prod_{h=1}^{l} |\mathcal{R}_p^{(h)}|$ blocks of order $\prod_{h=l+1}^{H} |\mathcal{R}_p^{(h)}|$ along the diagonal of $Q_D(p,p) + Q(p,p)_{DU(l)} + Q(p,p)_{DL(l)} + \omega Q(p,p)_{U(l)}$ for $p = 0, \ldots, N-1$. In [315], a recursive algorithm is given for a nonsingular linear system with a lower-triangular coefficient matrix in the

form of a sum of Kronecker products and a nonzero right-hand side. Such a system arises in *backward* SOR. Therein, a version of the same algorithm for backward BSOR is also discussed. Here we remark that a non-recursive block upper-triangular solution algorithm for (6.13) is also possible [58] and a block row-oriented version is preferable in the presence of functional transitions as in Algorithm 6.

Observe in Algorithm 6 that initially the nonzero right-hand side subvector $\mathbf{b}(\mathcal{R}_p)$ can be efficiently computed using the vector–Kronecker product multiplication algorithm, since $Q_{L(l)}$, $Q_{U(l)}$, $Q_{DL(l)}$, and $Q_{DU(l)}$ are expressed using sums of Kronecker products. Furthermore, $Q((i_1,\ldots,i_l),(i_1,\ldots,i_l))$ for $(i_1,\ldots,i_l) \in \times_{h=1}^{l} \mathcal{R}_p^{(h)}$ and $p = 0,\ldots,N-1$ is given in (6.12) in terms of a sum of Kronecker products, and $Q((i_1,\ldots,i_l),(j_1,\ldots,j_l))$ for $(j_1,\ldots,j_l) \in \times_{h=1}^{l} \mathcal{R}_p^{(h)}$ and $(j_1,\ldots,j_l) > (i_1,\ldots,i_l)$ can be expressed in terms of a sum of Kronecker products using (6.4) as

$$
\begin{aligned}
&Q((i_1,\ldots,i_l),(j_1,\ldots,j_l)) \\
&= \sum_{k\in K_{p,p}} \alpha_k \sum_{h=1}^{l} \left(\prod_{h'=1}^{h-1} d_k^{(h')}(i_{h'},i_{h'}) \right) u_k^{(h)}(i_h,j_h) \left(\prod_{h'=h+1}^{l} q_k^{(h')}(i_{h'},j_{h'}) \right) \\
&\qquad \left(\bigotimes_{h'=l+1}^{H} Q_k^{(h')}(\mathcal{R}_p^{(h)},\mathcal{R}_p^{(h)}) \right) .
\end{aligned}
$$

Algorithm 6. *Non-recursive block upper-triangular solution at level l in iteration it of BSOR for MCs based on Kronecker products.*

For reachable state space partition $p := 0,\ldots,N-1$,
$\mathbf{b}(\mathcal{R}_p) := (1-\omega)\left(\boldsymbol{\pi}_{(it)}(\mathcal{R}_p)Q_D(p,p) + \boldsymbol{\pi}_{(it)}(\mathcal{R}_p)Q(p,p)_{DU(l)} + \boldsymbol{\pi}_{(it)}(\mathcal{R}_p)Q(p,p)_{DL(l)}\right)$
$-\omega\left(\boldsymbol{\pi}_{(it)}(\mathcal{R}_p)Q(p,p)_{L(l)} + \sum_{w=p+1}^{N-1} \boldsymbol{\pi}_{(it)}(\mathcal{R}_w)Q(w,p) + \sum_{w=0}^{p-1} \boldsymbol{\pi}_{(it+1)}(\mathcal{R}_w)Q(w,p)\right)$;
For row of blocks $(i_1,\ldots,i_l) \in \times_{h=1}^{l} \mathcal{R}_p^{(h)}$ lexicographically,
Solve $\boldsymbol{\pi}_{(it+1)}((i_1,\ldots,i_l))Q((i_1,\ldots,i_l),(i_1,\ldots,i_l)) = \mathbf{b}((i_1,\ldots,i_l))$;
For column of blocks $(j_1,\ldots,j_l) \in \times_{h=1}^{l} \mathcal{R}_p^{(h)}$
such that $(j_1,\ldots,j_l) > (i_1,\ldots,i_l)$,
$\mathbf{b}((j_1,\ldots,j_l)) := \mathbf{b}((j_1,\ldots,j_l)) - \omega\boldsymbol{\pi}_{(it+1)}((i_1,\ldots,i_l))Q((i_1,\ldots,i_l),(j_1,\ldots,j_l))$.

To the contrary of BJOR, the nonsingular diagonal blocks $Q((i_1,\ldots,i_l),(i_1,\ldots,i_l))$ in BSOR must be solved in lexicographical order. If there is space, one can generate and factorize in sparse storage these blocks as in BJOR at the outset and solve the $\sum_{p=0}^{N-1} \prod_{h=1}^{l} |\mathcal{R}_p^{(h)}|$ systems directly at each iteration.

Otherwise, one can use an iterative method such as BSOR, since the off-diagonal parts of diagonal blocks are also sums of Kronecker products. After each block is solved for the unknown subvector $\boldsymbol{\pi}_{(it+1)}((i_1,\ldots,i_l))$, $\mathbf{b}(\mathcal{R}_p)$ is updated by multiplying the computed subvector with the corresponding row of blocks above the diagonal.

Note that when $N = 1$, hence, $p = 0$, the two summations on the right-hand side of $\mathbf{b}(\mathcal{R}_p)$ in Algorithm 6 disappear, and since there is a single reachable state space partition in this case, $\mathcal{R}_0$ can be dropped from the argument lists of $\boldsymbol{\pi}_{(it)}$ and $\mathbf{b}$; furthermore, $Q(0,0)_{L(l)}$ becomes $Q_{L(l)}$, implying the fourth term on the right-hand side of $\mathbf{b}$ reduces to $-\omega\boldsymbol{\pi}_{(it)}Q_{L(l)}$ as in Algorithm 2 of [101]. Finally, we emphasize that BSOR at level l reduces to SOR if $Q_{DL(l)} = 0$ (see Remark 4.1 in [315]).

The block iterative solvers, which are sometimes called *two-level* (or *two-stage*) iterative solvers [243], discussed in this section are coded within the `NSolve` package of the APNN toolbox [7, 22]. These solvers are shown to be more effective than point solvers on many test cases [58, 315]. Furthermore, to the contrary of block partitionings considered in [123] for sparse MCs, block partitionings of Kronecker products are nested and recursive due to the lexicographical ordering of states. Therefore, there tends to be more common structure among the diagonal blocks of the transition matrix of a MC expressed using sums of Kronecker products. Diagonal blocks having identical off-diagonal parts and diagonals which differ by a multiple of the identity [58] are exploited as discussed in Section 4.3. Such diagonal blocks can share and work with the real Schur factorization of only one diagonal block. This approach saves not only from time spent for factorization of diagonal blocks at the outset but also from space. The work in [58] also considers a three-level version of BSOR for MCs based on Kronecker products in which the diagonal blocks that are too large to be factorized are solved using BSOR. Similar results also appear in [172] for BGS. Finally, we remark that it is possible to alter the nonzero structure of the transition matrix underlying the Kronecker representation of a MC by reordering factors and states of factors so as to make it more suitable for block iterative methods. Power and JOR methods do not benefit from such reordering since the subdominant eigenvalues of their iteration matrices remain the same.

There are also *Schwarz* methods, which can be considered as a generalization of block iterative methods based on splittings in which the partitioning of the reachable state space $\mathcal{R}$ into N' subsets has overlaps (i.e., $\bigcup_{p=0}^{N'-1}\mathcal{G}_p = \mathcal{R}$ and $\mathcal{G}_p \cap \mathcal{G}_w \neq \varnothing$ for $p \neq w$). Their additive versions [42] become BJOR, and their multiplicative versions become BSOR when the overlaps are removed. Schwarz methods tend to accelerate the convergence of the corresponding block iterative methods, the amount of acceleration depending on the amount of overlap [234]. These methods are yet to be used with Kronecker-based Markovian representations.

The next section discusses various preconditioners to be used with projection methods for MCs based on Kronecker products.

6.2 Preconditioned Projection Methods

Projection methods for MCs [17, 123, 278, 305] are *non-stationary* iterative methods [168, 280] using a larger number of supplementary vectors than block iterative methods to expedite the solution process. The most commonly used projection methods for the solution of nonsymmetric linear systems are *bi-conjugate gradient* (BCG) [140], *generalized minimal residual* (GMRES) [281], *conjugate gradient squared* (CGS) [299], *quasi-minimal residual* (QMR) [149], and *bi-conjugate gradient stabilized* (BICGSTAB) [317]. Among these, GMRES uses as many supplementary vectors as the Krylov subspace [168, 280] size and therefore has the highest memory requirements.

Projection methods need to be used with preconditioners [28] to result in effective solvers. At each iteration of a preconditioned projection method, the row residual vector, $\mathbf{r}$, is used as the right-hand side of the linear system

$$\mathbf{z}M = \mathbf{r} \tag{6.14}$$

to compute the preconditioned row residual vector, $\mathbf{z}$. The objective of this preconditioning step is to improve the error in the approximate solution vector at that iteration. Note that if M were a multiple of I (as in (6.10)), the preconditioned residual would be a multiple of the residual computed at that iteration, implying no improvement. Hence, the preconditioner should approximate the coefficient matrix of the original system in a better way, yet the solution of linear systems as in (6.14) involving the *preconditioner* matrix, M, should be cheap. It is shown in [123] through a large number of numerical experiments on benchmark problems that, to result as effective solvers, projection methods for sparse MCs should be used with preconditioners, such as those based on *incomplete* LU (ILU) factorizations [123, 279] or explicit *approximate inverses* (AINV) [29]. However, it is still not clear how one can devise ILU- or AINV-type preconditioners for infinitesimal generators that are in the form of (2.10).

So far, various preconditioners are proposed for Kronecker structured representations such as those based on truncated Neumann series [305, 307], the cheap and separable preconditioner [50], circulant preconditioners for a specific class of problems [76], and the Kronecker sum preconditioner [312] which has been shown to work effectively on some small problems. The Kronecker product approximate preconditioner for MCs based on Kronecker products developed in a sequence of papers [210, 211, 212], although encouraging, is in the form of a prototype implementation. Numerical experiments in [50, 52, 211, 212, 307] indicate that there is still room for research regarding the development of effective preconditioners for MCs based on Kronecker products.

In introducing another class of preconditioners, we remark that each of the block iterative methods introduced in this work is actually a preconditioned power method for which the preconditioning matrix is M in (6.8). Since

M is based on Kronecker products, a BSOR preconditioner exploiting this property is proposed in [60]. To the contrary of the BSOR preconditioner entertained for sparse MCs in [123], the BSOR preconditioner for MCs based on Kronecker products has a rich structure induced by the lexicographical ordering of states. We provide the BSOR preconditioning step (6.14) using M^{BSOR} at level l for MCs based on Kronecker products in Algorithm 7. You will notice its resemblance to the BSOR iteration it at level l for MCs based on Kronecker products in Algorithm 6.

Projection methods and their preconditioned versions discussed in this section are coded within the `NSolve` package of the APNN toolbox [7, 22]. Through numerical experiments, it is shown in [60] that two-level BSOR preconditioned projection methods in which the diagonal blocks are solved exactly emerge as effective solvers that are competitive with block iterative methods and ML methods.

ALGORITHM 7. *BSOR preconditioning step at level l for MCs based on Kronecker products.*

For reachable state space partition $p := 0, \ldots, N-1$,
  $\mathbf{b}(\mathcal{R}_p) := \mathbf{r}(\mathcal{R}_p) - \sum_{w=0}^{p-1} \mathbf{z}(\mathcal{R}_w) Q(w,p)$;
  For row of blocks $(i_1, \ldots, i_l) \in \times_{h=1}^{l} \mathcal{R}_p^{(h)}$ lexicographically,
    Solve $\mathbf{z}((i_1, \ldots, i_l)) Q((i_1, \ldots, i_l), (i_1, \ldots, i_l)) = \mathbf{b}((i_1, \ldots, i_l))$;
    If $\omega \neq 1$,
      $\mathbf{z}((i_1, \ldots, i_l)) := \omega \mathbf{z}((i_1, \ldots, i_l))$;
    For column of blocks $(j_1, \ldots, j_l) \in \times_{h=1}^{l} \mathcal{R}_p^{(h)}$
    such that $(j_1, \ldots, j_l) > (i_1, \ldots, i_l)$,
      $\mathbf{b}((j_1, \ldots, j_l)) := \mathbf{b}((j_1, \ldots, j_l))$
        $-\mathbf{z}((i_1, \ldots, i_l)) Q((i_1, \ldots, i_l), (j_1, \ldots, j_l))$.

It will be interesting to compare JOR, BJOR, and SOR preconditioners as defined in (6.11) and (6.13) with existing preconditoners for MCs based on Kronecker products. Clearly, the class of ML methods proposed in [59] is another candidate for preconditioning projection methods.

In the next section, we introduce a simple version of the ML method [53, 59] for irreducible MCs based on Kronecker products which is intimately related to the well-known IAD method [74, 200, 305], but is not restricted to having two levels. A class of ML methods are then discussed in terms of the simple ML method.

6.3 Multilevel Methods

One of the most effective methods for computing the steady-state vector when the transition matrix of the MC is large and sparse is iterative aggregation–

disaggregation (IAD). To be able to use IAD, a partitioning of the reachable state space as in

$$\mathcal{R} = \bigcup_{p=0}^{N'-1} \mathcal{G}_p, \quad \mathcal{G}_p \cap \mathcal{G}_w = \varnothing \quad \text{for } p \neq w, \tag{6.15}$$

and a function $f : \mathcal{R} \mapsto \mathcal{N}'$ with $\mathcal{N}' = \{0, \ldots, N'-1\}$ that maps the detailed subset of states $\mathcal{G}_p$ to the coarser state $\{p\}$ for $p = 0, \ldots, N'-1$ needs to be identified up to a reordering and renumbering of the states.

Originally the IAD method [289] is devised for DTMCs. For a CTMC, the uniformized generator matrix P in (5.3) (cf. (1.2)) for steady-state analysis may be considered. Given $\boldsymbol{\pi}_{(0)} \in \mathbb{R}_{>0}^{1\times|\mathcal{R}|}$ such that $\boldsymbol{\pi}_{(0)}\mathbf{e} = 1$, IAD proceeds by performing computations at two different levels during iteration it. At the *coarse* level, *aggregation* is performed by mapping $\mathcal{G}_p$ to $\{p\}$ for $p = 0, \ldots, N'-1$, using $\boldsymbol{\pi}_{(it)}$ to define the aggregated system of linear equations

$$\mathbf{z}_{(it)}\, C_{\boldsymbol{\pi}_{(it)}} = \mathbf{z}_{(it)}, \quad \mathbf{z}_{(it)}\mathbf{e} = 1,$$

where

$$C_{\boldsymbol{\pi}_{(it)}} = \begin{pmatrix} c_{\boldsymbol{\pi}_{(it)}}(0,0) & \cdots & c_{\boldsymbol{\pi}^{(it)}}(0,N'-1) \\ \vdots & \ddots & \vdots \\ c_{\boldsymbol{\pi}_{(it)}}(N'-1,0) & \cdots & c_{\boldsymbol{\pi}_{(it)}}(N'-1,N'-1) \end{pmatrix},$$

and then the aggregated linear system is solved for $\mathbf{z}_{(it)} \in \mathbb{R}_{>0}^{1\times N'}$. To this end, the *aggregated* (or *coupling* [241]) *matrix* $C_{\boldsymbol{\pi}_{(it)}}$ needs to be an irreducible and aperiodic transition probability matrix satisfying $C_{\boldsymbol{\pi}_{(it)}} \in \mathbb{R}_{\geqslant 0}^{N'\times N'}$ and $C_{\boldsymbol{\pi}_{(it)}}\mathbf{e} = \mathbf{e}$. At the *fine* level, through *disaggregation* and a splitting-based iterative method, $\mathbf{z}_{(it)}$ is used to compute a hopefully better solution $\boldsymbol{\pi}_{(it+1)}$. The objective of IAD is to converge relatively fast to $\boldsymbol{\pi}$ with a reasonable accuracy.

Now, let us recall a result mentioned in Section 6.1. When none of the diagonal blocks in Q (for that matter, P) can be symmetrically permuted to block diagonal form, $T^{BJacobi}$ and T^{BGS} will have states of each partition in the same periodic class. A consequence of this often overlooked result is that regardless of the periodicity of the iteration matrix T, if such a block iteration is coupled with an aggregation step, *global convergence* of IAD is guaranteed [87].

Here, global convergence refers to the fact that

$$\lim_{it\to\infty} \|\boldsymbol{\pi}_{(it)} - \boldsymbol{\pi}\| = 0 \quad \text{for arbitrary} \quad \boldsymbol{\pi}_{(0)} \in \mathbb{R}_{>0}^{1\times|\mathcal{R}|}.$$

This is a stronger result than *local convergence* which requires that

$$\lim_{it\to\infty} \|\boldsymbol{\pi}_{(it)} - \boldsymbol{\pi}\| = 0 \quad \text{if} \quad \boldsymbol{\pi}_{(0)} \in \Omega(\boldsymbol{\pi}),$$

where $\Omega(\boldsymbol{\pi})$ is some neighborhood of $\boldsymbol{\pi}$.

For fast convergence, *nearly completely decomposable* (NCD) partitionings are considered for DTMCs in [200, 309, 321], which are inspired by earlier work [85, 295] related to NCDness. A block partitioning

$$P = E + \mathrm{diag}(P(0,0), \ldots, P(N'-1, N'-1))$$

is said to be NCD if $\|E\|_1$, that is, the *degree of coupling*, is relatively small with respect to 1. NCD partitionings are frequently met in practice due to the existence of different time scales in system models [308]. An NCD partitioning for a user-specified *decomposability parameter* $\gamma \in (0,1)$ can be obtained as discussed in [97]. When the block partitioning is NCD, $\mathbf{z}_{(it)}$ may need to be computed with the help of the *Grassmann–Taksar–Heyman (GTH)* idea [167, 293] which uses only positive floating-point arithmetic to achieve high accuracy as shown in [121]. That NCD partitioning-based IAD with BGS as the block iteration converges globally with *convergence factor* $\|E\|_1$ (where rate of convergence is the negated logarithm of convergence factor) [74, 237, 304] provides a very strong result for DTMCs. However, the block partitioning suggested by the Kronecker form may not be NCD. Therefore, IAD methods using non-NCD partitionings possibly with point iterative methods based on splittings need to be considered [77, 176, 182, 204, 244, 246, 316, 325]. This is something we investigate next so that it guides us in developing the ML method for Kronecker-based Markovian representations.

The local convergence proof of an IAD method for nonnegative matrices that uses the power method after disaggregation is provided in [221]. The particular IAD method computes the nonnegative solution vector of a problem that is intimately related to (6.9), but has a nonnegative consistent right-hand side vector, and for a given weight vector $\mathbf{w} \in \mathbb{R}_{>0}^{|\mathcal{R}|\times 1}$, the weighted ith row sum of the matrix at hand is less than $\mathbf{w}(i)$ for $i = 0, \ldots, |\mathcal{R}| - 1$. The local convergence result of the IAD method in [221] is extended to DTMCs in [222] and to inexact solution of the aggregated system in [231].

The *aggregation* (or *restriction*) *operator* $R \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}|\times N'}$ and the *disaggregation* (or *prolongation*) *operator* $S_{\boldsymbol{\pi}_{(it)}} \in \mathbb{R}_{\geqslant 0}^{N'\times|\mathcal{R}|}$ used in the IAD method for DTMCs are given entrywise in [224] as

$$r(i,p) = \begin{cases} \mathbf{w}(i) & \text{if } f(i) = p \\ 0 & \text{otherwise} \end{cases},$$

$$s_{\boldsymbol{\pi}_{(it)}}(p,i) = \begin{cases} \dfrac{\boldsymbol{\pi}_{(it)}(i)}{\sum_{j\in\mathcal{R},\ f(j)=p} \mathbf{w}(j)\boldsymbol{\pi}_{(it)}(j)} & \text{if } f(i) = p \\ 0 & \text{otherwise} \end{cases}$$

with a weight vector $\mathbf{w} \in \mathbb{R}_{>0}^{|\mathcal{R}|\times 1}$ as in [221] normally set to $\mathbf{e}$.

The disaggregation operator $S_{\boldsymbol{\pi}_{(it)}}$ depends on the current solution vector $\boldsymbol{\pi}_{(it)} \in \mathbb{R}_{>0}^{1\times|\mathcal{R}|}$ and has the same nonzero structure as R^T, implying

$$S_{\boldsymbol{\pi}_{(it)}} R = I.$$

When $\boldsymbol{\pi}_{(it)} \in \mathbb{R}_{>0}^{1\times|\mathcal{R}|}$, it is also possible to define the nonnegative *projector*

$$B_{\boldsymbol{\pi}_{(it)}} = RS_{\boldsymbol{\pi}_{(it)}}$$

which satisfies

$$B^2_{\boldsymbol{\pi}_{(it)}} = B_{\boldsymbol{\pi}_{(it)}}, \quad \boldsymbol{\pi}_{(it)}B_{\boldsymbol{\pi}_{(it)}} = \boldsymbol{\pi}_{(it)}, \quad \text{and} \quad B_{\boldsymbol{\pi}_{(it)}}\mathbf{w} = \mathbf{w}.$$

In terms of the aggregation and disaggregation operators, the aggregation step of IAD for DTMCs computes

$$C_{\boldsymbol{\pi}_{(it)}} := S_{\boldsymbol{\pi}_{(it)}}PR$$

and solves for $\mathbf{z}_{(it)} \in \mathbb{R}_{>0}^{1\times N'}$ in

$$\mathbf{z}_{(it)}C_{\boldsymbol{\pi}_{(it)}} = \mathbf{z}_{(it)}, \quad \mathbf{z}_{(it)}\mathbf{e} = 1,$$

whereas the disaggregation step followed by one power iteration is given as

$$\boldsymbol{\pi}_{(it+1)} := \mathbf{z}_{(it)}S_{\boldsymbol{\pi}_{(it)}}T \quad \text{with} \quad T = P.$$

A more general IAD method to compute the steady-state vector of DTMCs is provided in [225]. The method therein is shown to be globally convergent if a sufficiently high power of P is used in the aggregation step or a sufficiently large number of power iterations are performed on $\hat{P} = 0.5(I + P)$ after the disaggregation step. This IAD method is investigated further in a sequence of papers [223, 226] together with another intimately related IAD method devised for the singular version of the problem with the right-hand side in [221]. Letting

$$\Pi = \mathbf{e}\boldsymbol{\pi}$$

denote the square matrix with the steady-state vector $\boldsymbol{\pi}$ in its rows, that is, the part of P that corresponds to the eigenvalue 1, and using the *spectral decomposition* of P as in

$$P = \Pi + Z,$$

so that

$$\Pi^2 = \Pi, \quad \Pi Z = Z\Pi = 0, \quad \rho(Z) < 1, \quad \text{and} \quad \Pi(P - I) = 0,$$

it is shown that both IAD methods are locally convergent if a particular convergent matrix (dependent on the nonnegative projector $B_{\boldsymbol{\pi}_{(it)}}$, the iteration matrix T of the method used after disaggregation, and Z) exists. Furthermore, fast convergence of these IAD methods in one iteration is guaranteed when off-diagonal blocks in the strictly block lower-triangular part of T are outer products and therefore rank–1 [223].

The version of the IAD method with one power iteration taking place after disaggregation is shown in [228] to be locally convergent for $\delta \in (0, 1)$ with

convergence factor $(1-\delta)$ when $P \geqslant \delta\Pi$, or with convergence factor $\sqrt{1-\delta}$ when P has at least one positive column with 1-norm δ. Note that the former condition requires $P > 0$; therefore, both conditions are difficult to meet in practice. The same IAD method with P^{η_2} used in the aggregation step, η_1 power iterations (i.e., $T = P^{\eta_1}$) after the disaggregation step, and $\eta_1 \geqslant \eta_2$ is globally convergent for $\delta \in (0,1)$ when

$$P^{\eta_2} \geqslant \delta\Pi \quad \text{and} \quad \eta_1 \geqslant \eta_2 \frac{ln\delta - ln2}{ln(1-\delta)}.$$

We remark that since the power method is globally convergent for $\delta \in (0,1)$ with convergence factor $(1-\delta)$ when $P \geqslant \delta\Pi$, the IAD method with one power iteration taking place after disaggregation is not better than the power method and, therefore, cannot be recommended. Furthermore, IAD methods that are based on using powers of P during the aggregation step do not seem to be feasible in a sparse setting due to new nonzeros that would be introduced by this operation.

A class of IAD methods which use P in the aggregation step and a polynomial of P as the iteration matrix T of the method after disaggregation is considered in [269]. Therein, it is shown that IAD is locally convergent when $T = P$ and $p(i,i) > 0$ for $i = 0, \ldots, |\mathcal{R}| - 1$, or when $T = \alpha P + (1-\alpha)I$ for $\alpha \in (0,1)$, implying $T(i,i) > 0$ for $i = 0, \ldots, |\mathcal{R}| - 1$. In the latter case, the convergence factor is minimal for $\alpha \in (0.5, 1)$. In [270], the aperiodicity of an irreducible transition probability matrix associated with the *stochastic complement* [241] of $B_{\boldsymbol{\pi}_{(it)}}(p,p)P(p,p)$ for $p = 0, \ldots, N' - 1$ is provided as a necessary and sufficient condition for local convergence of IAD with one power iteration after disaggregation. The convergence factor of IAD for subvector p of $\boldsymbol{\pi}$ turns out to be the magnitude of the subdominant eigenvalue of this matrix. Many other interesting results are derived from experiments in [227, 271].

In the Kronecker-based representation of MCs, the partitioning (6.15) to be used by the IAD method can very well correspond to the block partitioning (2.10) at level 0 for which $N' = N$ and $\mathcal{G}_p = \mathcal{R}_p$ for $p = 0, \ldots, N-1$ when there are multiple reachable state space partitions (i.e., $N > 1$), or to another block partitioning at a higher level number (see, for instance, those in Table 4.1). It is even possible to use different level numbers at each reachable state space partition $\mathcal{R}_p$ to control the sizes of the diagonal blocks underlying the partitioning.

The challenge is to provide a scalable implementation of IAD. The first results along this direction for Kronecker-based representations of MCs couple aggregation–disaggregation steps with various iterative methods to accelerate convergence [43, 49, 50, 52]. An IAD method for MCs based on Kronecker products and its adaptive version, which analyzes aggregated systems for those parts where the error is estimated to be high, are proposed in [47] and [48], respectively. The adaptive IAD method in [48] is improved in [53] through a recursive definition and called ML. Here, we first present this method for generator matrices in the Kronecker form of (2.10).

The ML method is conceived so that aggregation of states takes place in a systematic manner, within each reachable state space partition independently of other reachable state space partitions. To keep the discussion simple, we assume that aggregation is performed in the order of submodel indices from 1 to H and disaggregation in the reverse order. At level 0, we have the unaggregated generator matrix Q. Therefore, at level 0 we aggregate submodel 1 and then move to level 1. At level l, we aggregate submodel $l+1$ since submodels 1 through l will have been aggregated previously. If we forget about the diagonal correction that sums the rows of the generator matrix to 0, term $k \in \mathcal{K}_{p,w}$ in aggregated block (p, w) at level l will correspond to the aggregation of the first $l+1$ submodels. In this way, it is possible to define an aggregated matrix at each partitioning level l for $l = 1, \ldots, H$ that is an $(N \times N)$ block matrix. Block (p, w) of this aggregated matrix can be expressed as a sum of Kronecker products with $|\mathcal{K}_{p,w}|$ terms as in (2.10) for $p, w = 0, \ldots, N-1$. Thus, it will be represented compactly by the multiplication of α_k with a diagonal $(\prod_{h=l+2}^{H} |\mathcal{R}_p^{(h)}| \times \prod_{h=l+2}^{H} |\mathcal{R}_p^{(h)}|)$ matrix describing the effect of aggregating the first $l+1$ submodels on the remaining $H-l-1$ unaggregated submodels, which in turn is multiplied by the Kronecker product $\bigotimes_{h=l+2}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$ corresponding to the unaggregated $H-l-1$ submodels.

This approach requires us to first associate with the H-tuple state $\mathbf{i} \in \mathcal{R}_p$ the partition number p to which it belongs. We need this simply because, after aggregation of a particular submodel, it is possible for states in different reachable state space partitions to map to the same aggregated state; but, as motivated in the previous paragraph, each aggregated state must remain in its original partition. This can be done as in [61] by inserting the partition number p as the $(H+1)$st entry at the end of the H-tuple representing state $\mathbf{i} \in \mathcal{R}_p$ so that states are represented by $(H+1)$-tuples or, for instance, by using a subscript p with $\mathbf{i} \in \mathcal{R}_p$ as we will do here. Second, a level number needs to be associated with reachable state space partition $\mathcal{R}_p$ and the aggregated states $\mathbf{i} \in \mathcal{R}_p$ for $p = 0, \ldots, N-1$ to specify the aggregated operator using sums of Kronecker products at each level.

So, let $\mathbf{i}_{(p,l)} \in \mathcal{R}_{p,l}$, where

$$\mathcal{R}_{p,l} = \bigtimes_{h=l+1}^{H} \mathcal{R}_p^{(h)} \quad \text{for } p = 0, \ldots, N-1 \text{ and } l = 0, \ldots, H-1$$

denotes the aggregated reachable state space partition p at level l in which submodels 1 through l are aggregated, with

$$\mathcal{R}_{p,0} = \mathcal{R}_p \quad \text{and} \quad \mathcal{R}_{p,H} = \{p\},$$

and the mapping

$$f_{(p,l)} : \mathcal{R}_{p,l} \mapsto \mathcal{R}_{p,l+1}$$

represents the aggregation of submodel $l+1$ with state space $\mathcal{R}_p^{(l+1)}$ for $p = 0, \ldots, N-1$ so that state $\mathbf{i}_{(p,l)} \in \mathcal{R}_{p,l}$ is mapped to state $\mathbf{i}_{(p,l+1)} \in \mathcal{R}_{p,l+1}$.

Furthermore, let the aggregated generator matrices $\tilde{Q}_{(it,l)}$ with aggregated reachable state space partitions $\mathcal{R}_{p,l}$, $p = 0, \ldots, N-1$, be defined at levels $l = 1, \ldots, H$ with $\tilde{Q}_{(it,0)} = Q$ for iteration it. Observe that $\tilde{Q}_{(it,l)}$ is an $(\sum_{p=0}^{N-1} |\mathcal{R}_{p,l}| \times \sum_{p=0}^{N-1} |\mathcal{R}_{p,l}|)$ matrix. Finally, let the power method be used as *smoother* (or *accelerator*) before aggregation $\eta_{(it,l)}$ times and after disaggregation $\nu_{(it,l)}$ times with the uniformization parameter

$$\Gamma_{(it,l)} \in \Big[\max_{\mathbf{i}_{(p,l)} \in \mathcal{R}_{p,l},\ p=0,\ldots,N-1} |\tilde{q}_{(it,l)}(\mathbf{i}_{(p,l)}, \mathbf{i}_{(p,l)})|, \infty \Big)$$

at level l for iteration it.

Then the ML iteration matrix that facilitates the ML iteration

$$\boldsymbol{\pi}_{(it+1,l)} := \boldsymbol{\pi}_{(it,l)} T^{ML}_{(it,l)} \quad \text{for } it = 0, \ldots, \texttt{maxit} - 1$$

at level l for iteration it is given by

$$T^{ML}_{(it,l)} = \left(I + \frac{1}{\Gamma_{(it,l)}} \tilde{Q}_{(it,l)} \right)^{\eta_{(it,l)}} R_{(l)} T^{ML}_{(it,l+1)} S_{\mathbf{x}_{(it,l)}} \left(I + \frac{1}{\Gamma_{(it,l)}} \tilde{Q}_{(it,l)} \right)^{\nu_{(it,l)}}. \tag{6.16}$$

Note that the definition of $T^{ML}_{(it,l)}$ is recursive, and to the contrary of block iterative methods, the ML iteration matrix in (6.16) changes from iteration to iteration, and hence, the method is non-stationary.

At iteration it, the recursion ends and backtracking starts when $\tilde{Q}_{(it,l+1)}$ is the last aggregated generator matrix and solved to give

$$T^{ML}_{(it,l+1)} = \mathbf{e}\boldsymbol{\pi}_{(it+1,l+1)}, \quad \text{where } \boldsymbol{\pi}_{(it+1,l+1)} \tilde{Q}_{(it,l+1)} = \mathbf{0} \text{ and } \boldsymbol{\pi}_{(it+1,l+1)} \mathbf{e} = 1.$$

The level to end recursion depends on available memory since there must be space to store and factorize the aggregated generator matrix at that level if a direct method is employed for an accurate solution. When Q is irreducible and $\boldsymbol{\pi}_{(0,0)} \in \mathbb{R}^{1 \times |\mathcal{R}|}_{>0}$, the aggregated generator matrices $\tilde{Q}_{(it,l+1)}$ are irreducible [53, 61], and the ML method has been observed to converge if a sufficient number of smoothings are performed to improve the approximate solution vector $\boldsymbol{\pi}_{(it,l)}$ at each level.

Clearly, the implementation of the ML method with the iteration matrix in (6.16) should not require the explicit generation and storage of the aggregated generator matrices $\tilde{Q}_{(it,l)}$ for MCs based on Kronecker products. To the contrary, as other iterative methods before, it should rely on vector–Kronecker product multiplications with smaller matrices and some vector operations.

The pre-smoothed vector in (6.16) is obtained from

$$\mathbf{x}_{(it,l)} := \boldsymbol{\pi}_{(it,l)} \left(I + \frac{1}{\Gamma_{(it,l)}} \tilde{Q}_{(it,l)} \right)^{\eta_{(it,l)}}. \tag{6.17}$$

The aggregation operator $R_{(l)} \in \mathbb{R}_{\geqslant 0}^{\sum_{p=0}^{N-1} |\mathcal{R}_{p,l}| \times \sum_{p=0}^{N-1} |\mathcal{R}_{p,l+1}|}$ and the disaggregation operator $S_{\mathbf{x}_{(m,l)}} \in \mathbb{R}^{\sum_{p=0}^{N-1} |\mathcal{R}_{p,l+1}| \times \sum_{p=0}^{N-1} |\mathcal{R}_{p,l}|}$ are given entrywise in [61] as

$$r_{(l)}(\mathbf{i}_{(p,l)}, \mathbf{i}_{(p,l+1)}) = \begin{cases} 1 & \text{if } f_{(p,l)}(\mathbf{i}_{(p,l)}) = \mathbf{i}_{(p,l+1)} \\ 0 & \text{otherwise} \end{cases} \tag{6.18}$$

and

$$\begin{aligned} s_{\mathbf{x}_{(it,l)}}(\mathbf{i}_{(p,l+1)}, \mathbf{i}_{(p,l)}) &= \frac{\mathbf{x}_{(it,l)}(\mathbf{i}_{(p,l)})}{\sum\limits_{\mathbf{j}_{(p,l)} \in \mathcal{R}_{p,l},\ f_{(p,l)}(\mathbf{j}_{(p,l)}) = \mathbf{i}_{(p,l+1)}} \mathbf{x}_{(it,l)}(\mathbf{j}_{(p,l)})} \\ &\qquad \text{if } f_{(p,l)}(\mathbf{i}_{(p,l)}) = \mathbf{i}_{(p,l+1)}, \\ s_{\mathbf{x}_{(it,l)}}(\mathbf{i}_{(p,l+1)}, \mathbf{i}_{(p,l)}) &= 0 \quad \text{otherwise} \end{aligned} \tag{6.19}$$

$$\text{for } \mathbf{i}_{(p,l)} \in \mathcal{R}_{p,l},\ \mathbf{i}_{(p,l+1)} \in \mathcal{R}_{p,l+1},\ \text{and } p = 0, \ldots, N-1\ .$$

Observe that both operators at level l are $(N \times N)$ block diagonal due to our choice to aggregate states only within each partition $\mathcal{R}_{p,l}$ for $p = 0, \ldots, N-1$.

The aggregation operator $R_{(l)}$ in (6.18) is defined by $f_{(p,l)} : \mathcal{R}_{p,l} \mapsto \mathcal{R}_{p,l+1}$ for $p = 0, \ldots, N-1$, and therefore is constant and need not be stored. At level l, the $|\mathcal{R}_{p,l}| = \prod_{h=l+1}^{H} |\mathcal{R}_p^{(h)}|$ states represented by $(H-l)$-tuples are mapped to the $|\mathcal{R}_{p,l+1}| = \prod_{h=l+2}^{H} |\mathcal{R}_p^{(h)}|$ states represented by $(H-l-1)$-tuples by $f_{(p,l)}$ through aggregation of the leading dimension $\mathcal{R}_p^{(l+1)}$ in $\mathcal{R}_{p,l}$ for $p = 0, \ldots, N-1$. We remark that this corresponds to an aggregation based on a contiguous and non-interleaved block partitioning if the states in $\mathcal{R}_{p,l}$ were ordered anti-lexicographically. On the other hand, the disaggregation operator $S_{\mathbf{x}_{(it,l)}}$ in (6.19) depends on the smoothed vector $\mathbf{x}_{(it,l)}$ in (6.17) and has the nonzero structure of $R_{(l)}^T$. Therefore, $S_{\mathbf{x}_{(it,l)}}$ can be stored in a vector of length $\sum_{p=0}^{N-1} |\mathcal{R}_{p,l}|$ since it has one nonzero per column by definition. These vectors amount to a total storage of $\sum_{l=0}^{H-1} \sum_{p=0}^{N-1} |\mathcal{R}_{p,l}|$ floating-point values if the recursion terminates at level H.

The aggregation operator $R_{(l)}$ and the disaggregation operator $S_{\mathbf{x}_{(it,l)}}$ have the same properties of the two operators used in the IAD method for DTMCs in this section. Similarly, it is also possible to define the nonnegative projector $B_{\mathbf{x}_{(it,l)}} \in \mathbb{R}_{\geqslant 0}^{\sum_{p=0}^{N-1} |\mathcal{R}_{p,l}| \times \sum_{p=0}^{N-1} |\mathcal{R}_{p,l}|}$ at level l as

$$B_{\mathbf{x}_{(it,l)}} = R_{(l)} S_{\mathbf{x}_{(it,l)}} \quad \text{for } l = 0, \ldots, H-1.$$

In the ML method, the pre-smoothed vector is aggregated using

$$\boldsymbol{\pi}_{(it,l+1)} := \mathbf{x}_{(it,l)} R_{(l)} \tag{6.20}$$

and passed to level $l+1$, which has the aggregated generator matrix

$$\tilde{Q}_{(it,l+1)} = S_{\mathbf{x}_{(it,l)}} \tilde{Q}_{(it,l)} R_{(l)}\ . \tag{6.21}$$

In [61], it is shown that $\tilde{Q}_{(it,l+1)}$ is a block matrix that can be expressed using sums of Kronecker products as in (2.10) with $\sum_{p=0}^{N-1}\sum_{w=0}^{N-1}|\mathcal{K}_{p,w}|$ *aggregation vectors* named $\mathbf{a}$ each of length at most $\max_{p\in\{0,\ldots,N-1\}}(\prod_{h=l+2}^{H}|\mathcal{R}_p^{(h)}|)$ and the submatrices corresponding to the factors $(l+2)$ through H.

More specifically, the $\mathbf{i}_{(p,l+1)}$st entry of the aggregation vector corresponding to the kth term in the Kronecker representation at level $(l+1)$ for block (p,w) at iteration it with $\mathbf{i}_{(p,l+1)} \in \mathcal{R}_{p,l+1}$ and $k \in \mathcal{K}_{p,w}$ is computed as

$$\mathbf{a}_{(it,l+1,p,w,k)}(\mathbf{i}_{(p,l+1)}) \tag{6.22}$$

$$:= \frac{\sum_{\mathbf{j}_{(p,l)}\in\mathcal{R}_{p,l},\ f_{(p,l)}(\mathbf{j}_{(p,l)})=\mathbf{i}_{(p,l+1)}} \mathbf{x}_{(it,l)}(\mathbf{j}_{(p,l)})\ \mathbf{a}_{(it,l,p,w,k)}(\mathbf{j}_{(p,l)})\ row\_sum}{\boldsymbol{\pi}_{(it,l+1)}(\mathbf{i}_{(p,l+1)})},$$

where

$$row\_sum := \mathbf{e}^T_{\mathbf{j}_{(p,l)}(l+1)} Q_k^{(l+1)}(\mathcal{R}_p^{(l+1)}, \mathcal{R}_w^{(l+1)})\mathbf{e}$$

yields the sum of entries in row $\mathbf{j}_{(p,l)}(l+1) \in \mathcal{R}_p^{(l+1)}$ of $Q_k^{(l+1)}(\mathcal{R}_p^{(l+1)}, \mathcal{R}_w^{(l+1)})$ with $\mathbf{e}_{\mathbf{j}_{(p,l)}(l+1)}$ being the $\mathbf{j}_{(p,l)}(l+1)$st column of I of order $|\mathcal{R}_p^{(l+1)}|$. At level 0, we set

$$\mathbf{a}_{(it,0,p,w,k)} := \mathbf{e}.$$

Then block (p,w) of $\tilde{Q}_{(it,l+1)}$ for $p,w = 0,\ldots,N-1$ can be expressed as

$$\tilde{Q}_{(it,l+1)}(p,w) = \begin{cases} \sum_{k\in\mathcal{K}_{p,w}} \tilde{Q}_{(it,l+1,k)}(p,w) + \tilde{Q}_{(it,l+1,D)}(p,p) & \text{if } p = w \\ \sum_{k\in\mathcal{K}_{p,w}} \tilde{Q}_{(it,l+1,k)}(p,w) & \text{otherwise} \end{cases},$$

where

$$\tilde{Q}_{(it,l+1,k)}(p,w) = \alpha_k\ \mathrm{diag}(\mathbf{a}_{(it,l+1,p,w,k)}) \bigotimes_{h=l+2}^{H} Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)}),$$

$$\tilde{Q}_{(it,l+1,D)}(p,p) = -\sum_{w=0}^{N-1}\sum_{k\in\mathcal{K}_{p,w}} \alpha_k\ \mathrm{diag}(\mathbf{a}_{(it,l+1,p,w,k)}) \bigotimes_{h=l+2}^{H} \mathrm{diag}(Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}). \tag{6.23}$$

Observe that $\tilde{Q}_{(it,l+1,D)}(p,p)$ returns a diagonal matrix with negative diagonal entries so that

$$\sum_{w=0}^{N-1} \tilde{Q}_{(it,l+1)}(p,w)\mathbf{e} = \mathbf{0} \quad \text{for } p = 0,\ldots,N-1.$$

In other words, row sums of the row of blocks corresponding to aggregated reachable state space partition p at level $l+1$, $\mathcal{R}_{p,l+1}$, of $\tilde{Q}_{(it,l+1)}$ must be equal to the $\mathbf{0}$ vector. If the recursion ends at level H, then $\tilde{Q}_{(it,H)}$ is an $(N \times N)$ generator matrix and therefore is generated and stored explicitly in row sparse format when there are multiple reachable state space partitions (i.e., $N > 1$) so that it can be solved either directly (e.g., by Gaussian elimination) if N is small, else iteratively using the smoother and the current approximation $\boldsymbol{\pi}_{(it,H)}$ as the starting vector. When $N = 1$, $\tilde{Q}_{(it,H)} = 0$, and hence, $\boldsymbol{\pi}_{(it+1,H)}$ can be set to 1 to start backtracking from the recursion.

The aggregation vectors $\mathbf{a}_{(it,l,p,w,k)}$ for $k \in \mathcal{K}_{p,w}$ at $l = 0$ by definition consist of all 1's, and therefore need not be stored. Furthermore, the computation of $\mathbf{a}_{(it,l+1,p,w,k)}$ in (6.22) suggests that if $\mathbf{a}_{(it,l,p,w,k)} = \mathbf{e}$ and row_sum evaluates to 1 for all $\mathbf{i}_{(p,l+1)} \in \mathcal{R}_{p,l+1}$, then $\mathbf{a}_{(it,l+1,p,w,k)} = \mathbf{e}$, since in this case the summation in the numerator will evaluate to the value in the denominator by the definition of the aggregation operator $R_{(l)}$. This is possible, for instance, when $Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_w^{(h)})$ for $h = 1, \ldots, l+1$ are all diagonal matrices of size $(|\mathcal{R}_p^{(h)}| \times |\mathcal{R}_w^{(h)}|)$ with 1's along their diagonal. Since submodel matrices forming $\tilde{Q}_{(it,l+1)}(p,w)$ for $p \neq w$ can very well be rectangular, we refrain from using I and remark that such aggregation vectors need not be stored either. Savings are also possible for those $k \in \mathcal{K}_{p,p}$ that have a single $Q_k^{(h)}(\mathcal{R}_p^{(h)}, \mathcal{R}_p^{(h)}) \neq I$ for $h = 1, \ldots, H$. In this case, the contribution of $\mathbf{a}_{(it,l+1,p,p,k)}$ can only be to the diagonal of $\tilde{Q}_{(it,l+1)}(p,p)$ and its effect will be canceled with the corresponding diagonal correction due to $\tilde{Q}_{(it,l+1,D)}(p,p)$. This implies that setting $\mathbf{a}_{(it,l+1,p,p,k)} = \mathbf{e}$ and not storing it will not change the result in this case as well.

The aggregation vectors for block (p,w) at a particular level have the same length but vary in length from $\prod_{h=2}^{H} |\mathcal{R}_p^{(h)}|$ at level 1 to $|\mathcal{R}_p^{(H)}|$ at level $(H-1)$, implying a storage requirement of at most

$$\sum_{p=0}^{N-1} \sum_{w=0}^{N-1} |\mathcal{K}_{p,w}| \sum_{l=1}^{H-1} \prod_{h=l+1}^{H} |\mathcal{R}_p^{(h)}|$$

floating-point values to facilitate the Kronecker representation of aggregated generator matrices. We remark that grouping of factors will further reduce the storage requirement for aggregation vectors.

Example 2. (ctnd.) Consider our three-dimensional problem with the parameter set

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6) = (\lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2, \mu_3) = (1, 2, 3, 2, 4, 6),$$

the initial distribution $\boldsymbol{\pi}_{(0,0)} = \mathbf{e}/12$, $\Gamma_{(0,0)} = 14$, and $\eta_{(0,0)} = \nu_{(0,0)} = 1$. Since

$$\tilde{Q}_{(0,0)} = \begin{array}{c} \\ 0\,0\,0 \\ 0\,0\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 2\,0\,0 \\ 2\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,1\,0 \\ 1\,1\,1 \\ 0\,0\,2 \\ 1\,0\,2 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 2 \end{array} \\ \left(\begin{array}{cccccccccccc} -9 & 6 & 1 & & & & 2 & & & & & \\ 6 & -12 & & 1 & & & & 2 & & & 3 & \\ 2 & & -11 & 6 & 1 & & & & 2 & & & \\ & 2 & 6 & -14 & & 1 & & & & 2 & & 3 \\ \hline & & 2 & & -2 & & & & & & & \\ & & & 2 & 6 & -8 & & & & & & \\ \hline 4 & & & & & & -4 & & & & & \\ & 4 & & & & & 6 & -10 & & & & \\ & & 4 & & & & 2 & & -6 & & & \\ & & & 4 & & & & 2 & 6 & -12 & & \\ \hline & 6 & & & & & & & & & -6 & \\ & & & 6 & & & & & & & 2 & -8 \end{array}\right) \end{array},$$

$\mathbf{x}_{(0,0)} = \boldsymbol{\pi}_{(0,0)}(I + \tilde{Q}_{(0,0)}/14)$ from (6.17) yields

$$\mathbf{x}_{(0,0)} = \left(\frac{17}{168}, \frac{20}{168}, \frac{16}{168}, \frac{19}{168}, \frac{19}{168}, \frac{7}{168}, \frac{20}{168}, \frac{8}{168}, \frac{16}{168}, \frac{4}{168}, \frac{13}{168}, \frac{9}{168}\right).$$

In this example, there are $N = 4$ reachable state space partitions, and hence, the aggregation operator in (6.18) and the disaggregation operator in (6.19) are block diagonal matrices. At level 0, they are, respectively, given by

$$R_{(0)} = \begin{array}{c} \\ 0\,0\,0 \\ 0\,0\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 2\,0\,0 \\ 2\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,1\,0 \\ 1\,1\,1 \\ 0\,0\,2 \\ 1\,0\,2 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 2 \end{array} \\ \left(\begin{array}{cccccccc} 1 & & & & & & \\ & 1 & & & & & \\ 1 & & & & & & \\ & 1 & & & & & \\ \hline & & 1 & & & & \\ & & & 1 & & & \\ \hline & & & & 1 & & \\ & & & & & 1 & \\ & & & & 1 & & \\ & & & & & 1 & \\ \hline & & & & & & 1 \\ & & & & & & 1 \end{array}\right) \end{array}$$

and

$$S_{\mathbf{x}_{(0,0)}} = \begin{array}{c} \\ 0\,0 \\ 0\,1 \\ 0\,0 \\ 0\,1 \\ 1\,0 \\ 1\,1 \\ 0\,2 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 2 \end{array} \\ \left(\begin{array}{cccccccccccc} \frac{17}{33} & & & \frac{16}{33} & & & & & & & & \\ & \frac{20}{39} & & & \frac{19}{39} & & & & & & & \\ & & & & & 1 & & & & & & \\ & & & & & & 1 & & & & & \\ & & & & & & & \frac{20}{36} & & \frac{16}{36} & & \\ & & & & & & & & \frac{8}{12} & & \frac{4}{12} & \\ & & & & & & & & & & \frac{13}{22} & \frac{9}{22} \end{array} \right) \end{array}.$$

The 12 states represented by 3-tuples in $\mathcal{R}$ are mapped to the 7 states represented by 2-tuples in $\mathcal{R}_{0,1}$, $\mathcal{R}_{1,1}$, $\mathcal{R}_{2,1}$, and $\mathcal{R}_{3,1}$. For instance, states $(0,0,0)$ and $(1,0,0)$ in $\mathcal{R}_{0,0}$ are mapped to $(0,0)$ in $\mathcal{R}_{0,1}$, whereas states $(0,0,2)$ and $(1,0,2)$ in $\mathcal{R}_{3,0}$ are mapped to $(0,2)$ in $\mathcal{R}_{3,1}$. Note that in this example, aggregation of submodel 1 at level 0 does not yield a reduction in the number of states in reachable state space partition $\mathcal{R}_{1,1}$ because $|\mathcal{R}_1^{(1)}| = 1$.

Using $R_{(0)}$ in (6.20), we obtain the starting approximation at level 1 as

$$\boldsymbol{\pi}_{(0,1)} = \left(\frac{33}{168}, \frac{39}{168}, \frac{19}{168}, \frac{7}{168}, \frac{36}{168}, \frac{12}{168}, \frac{22}{168} \right).$$

Through the interaction matrix

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \left(\begin{array}{cccc} \{1,3,4,6\} & \{1\} & \{2\} & \{3\} \\ \{4\} & \{6\} & & \\ \{5\} & & \{4,6\} & \\ \{6\} & & & \{4\} \end{array} \right) \end{array},$$

(cf. (2.8)), mapping of submodel states to reachable state space partitions given in Table 2.1, submodel 1's transition submatrices

$$Q_1^{(1)}(\mathcal{R}_0,\mathcal{R}_0) = \begin{pmatrix} & 1 \end{pmatrix}, \quad Q_3^{(1)}(\mathcal{R}_0,\mathcal{R}_0) = I_2, \quad Q_4^{(1)}(\mathcal{R}_0,\mathcal{R}_0) = \begin{pmatrix} 1 & \end{pmatrix},$$

$$Q_6^{(1)}(\mathcal{R}_0,\mathcal{R}_0) = I_2, \quad Q_1^{(1)}(\mathcal{R}_0,\mathcal{R}_1) = \begin{pmatrix} 1 \end{pmatrix}, \quad Q_2^{(1)}(\mathcal{R}_0,\mathcal{R}_2) = I_2,$$

$$Q_3^{(1)}(\mathcal{R}_0,\mathcal{R}_3) = I_2, \quad Q_4^{(1)}(\mathcal{R}_1,\mathcal{R}_0) = \begin{pmatrix} & 1 \end{pmatrix}, \quad Q_6^{(1)}(\mathcal{R}_1,\mathcal{R}_1) = I_1,$$

$$Q_5^{(1)}(\mathcal{R}_2,\mathcal{R}_0) = I_2, \quad Q_4^{(1)}(\mathcal{R}_2,\mathcal{R}_2) = \begin{pmatrix} 1 & \end{pmatrix}, \quad Q_6^{(1)}(\mathcal{R}_2,\mathcal{R}_2) = I_2,$$

$$Q_6^{(1)}(\mathcal{R}_3,\mathcal{R}_0) = I_2, \quad Q_4^{(1)}(\mathcal{R}_3,\mathcal{R}_3) = \begin{pmatrix} 1 & \end{pmatrix},$$

and the definition of aggregation vectors in (6.22), the 14 vectors used to represent the aggregated generator matrix $\tilde{Q}_{(0,1)}$ at level 1 are computed as

$$\begin{aligned}
\mathbf{a}_{(0,1,0,0,1)} &= \left(\frac{17}{33}, \frac{20}{39}\right)^T, \quad \mathbf{a}_{(0,1,0,0,4)} = \mathbf{a}_{(0,1,0,1,1)} = \left(\frac{16}{33}, \frac{19}{39}\right)^T, \\
\mathbf{a}_{(0,1,2,2,4)} &= \left(\frac{16}{36}, \frac{4}{12}\right)^T, \quad \mathbf{a}_{(0,1,3,3,4)} = \left(\frac{9}{22}\right), \\
\mathbf{a}_{(0,1,0,0,3)} &= \mathbf{a}_{(0,1,0,0,6)} = \mathbf{a}_{(0,1,0,2,2)} = \mathbf{a}_{(0,1,0,3,3)} = \mathbf{a}_{(0,1,1,0,4)} \\
&= \mathbf{a}_{(0,1,1,1,6)} = \mathbf{a}_{(0,1,2,0,5)} = \mathbf{a}_{(0,1,2,2,6)} = (1,1)^T, \\
\mathbf{a}_{(0,1,3,0,6)} &= (1).
\end{aligned}$$

Note that nine of the aggregation vectors are equal to $\mathbf{e}$ and need not be stored. All of these nine vectors except $\mathbf{a}_{(0,1,1,0,4)}$ are equal to $\mathbf{e}$ since $\mathbf{a}_{(0,0,p,w,k)} = \mathbf{e}$ and submodel 1's transition submatrices $Q_k^{(1)}(\mathcal{R}_p^{(1)}, \mathcal{R}_w^{(1)})$ for the respective transitions k are all diagonal matrices with 1's along their diagonal. For $\mathbf{a}_{(0,1,1,0,4)}$, the situation is slightly different. The submatrix used in the computation of $\mathbf{a}_{(0,1,1,0,4)}$ is given by $Q_4^{(1)}(\mathcal{R}_1^{(1)}, \mathcal{R}_0^{(1)}) = (0,1)$. Because $\mathbf{a}_{(0,0,1,0,4)} = \mathbf{e}$ and submatrix $Q_4^{(1)}(\mathcal{R}_1^{(1)}, \mathcal{R}_0^{(1)})$ is a row vector having a single nonzero with value 1, $\mathbf{a}_{(0,1,1,0,4)}$ also turns out to be equal to $\mathbf{e}$.

Using the aggregation vectors, the ten nonzero blocks of the aggregated generator matrix $\tilde{Q}_{(0,1)}$ are expressed as

$$\begin{aligned}
\tilde{Q}_{(0,1)}(0,0) &= \sum_{k\in\{1,3,4,6\}} \alpha_k \operatorname{diag}(\mathbf{a}_{(0,1,0,0,k)}) \bigotimes_{h=2}^{3} Q_k^{(h)}(\mathcal{R}_0^{(h)}, \mathcal{R}_0^{(h)}) \\
&\quad - \sum_{w=0}^{3} \sum_{k\in\mathcal{K}_{0,w}} \alpha_k \operatorname{diag}(\mathbf{a}_{(0,1,0,w,k)}) \bigotimes_{h=2}^{3} \operatorname{diag}(Q_k^{(h)}(\mathcal{R}_0^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}), \\
\tilde{Q}_{(0,1)}(0,1) &= \alpha_1 \operatorname{diag}(\mathbf{a}_{(0,1,0,1,1)}) \bigotimes_{h=2}^{3} Q_1^{(h)}(\mathcal{R}_0^{(h)}, \mathcal{R}_1^{(h)}), \\
\tilde{Q}_{(0,1)}(0,2) &= \alpha_2 \operatorname{diag}(\mathbf{a}_{(0,1,0,2,2)}) \bigotimes_{h=2}^{3} Q_2^{(h)}(\mathcal{R}_0^{(h)}, \mathcal{R}_2^{(h)}), \\
\tilde{Q}_{(0,1)}(0,3) &= \alpha_3 \operatorname{diag}(\mathbf{a}_{(0,1,0,3,3)}) \bigotimes_{h=2}^{3} Q_3^{(h)}(\mathcal{R}_0^{(h)}, \mathcal{R}_3^{(h)}), \\
\tilde{Q}_{(0,1)}(1,0) &= \alpha_4 \operatorname{diag}(\mathbf{a}_{(0,1,1,0,4)}) \bigotimes_{h=2}^{3} Q_4^{(h)}(\mathcal{R}_1^{(h)}, \mathcal{R}_0^{(h)}), \\
\tilde{Q}_{(0,1)}(1,1) &= \alpha_6 \operatorname{diag}(\mathbf{a}_{(0,1,1,1,6)}) \bigotimes_{h=2}^{3} Q_6^{(h)}(\mathcal{R}_1^{(h)}, \mathcal{R}_1^{(h)}), \\
&\quad - \sum_{w=0}^{1} \sum_{k\in\mathcal{K}_{1,w}} \alpha_k \operatorname{diag}(\mathbf{a}_{(0,1,1,w,k)}) \bigotimes_{h=2}^{3} \operatorname{diag}(Q_k^{(h)}(\mathcal{R}_1^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}),
\end{aligned}$$

$$\tilde{Q}_{(0,1)}(2,0) = \alpha_5 \ \mathrm{diag}(\mathbf{a}_{(0,1,2,0,5)}) \bigotimes_{h=2}^{3} Q_5^{(h)}(\mathcal{R}_2^{(h)}, \mathcal{R}_0^{(h)}) \ ,$$

$$\tilde{Q}_{(0,1)}(2,2) = \sum_{k\in\{4,6\}} \alpha_k \ \mathrm{diag}(\mathbf{a}_{(0,1,2,2,k)}) \bigotimes_{h=2}^{3} Q_k^{(h)}(\mathcal{R}_2^{(h)}, \mathcal{R}_2^{(h)})$$
$$- \sum_{w\in\{0,2\}} \sum_{k\in\mathcal{K}_{2,w}} \alpha_k \ \mathrm{diag}(\mathbf{a}_{(0,1,2,w,k)}) \bigotimes_{h=2}^{3} \mathrm{diag}(Q_k^{(h)}(\mathcal{R}_2^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}) \ ,$$

$$\tilde{Q}_{(0,1)}(3,0) = \alpha_6 \ \mathrm{diag}(\mathbf{a}_{(0,1,3,0,6)}) \bigotimes_{h=2}^{3} Q_6^{(h)}(\mathcal{R}_3^{(h)}, \mathcal{R}_0^{(h)}) \ ,$$

$$\tilde{Q}_{(0,1)}(3,3) = \alpha_4 \ \mathrm{diag}(\mathbf{a}_{(0,1,3,3,4)}) \bigotimes_{h=2}^{3} Q_4^{(h)}(\mathcal{R}_3^{(h)}, \mathcal{R}_3^{(h)})$$
$$- \sum_{w\in\{0,3\}} \sum_{k\in\mathcal{K}_{3,w}} \alpha_k \ \mathrm{diag}(\mathbf{a}_{(0,1,3,w,k)}) \bigotimes_{h=2}^{3} \mathrm{diag}(Q_k^{(h)}(\mathcal{R}_3^{(h)}, \mathcal{R}_w^{(h)})\mathbf{e}) \ .$$

Regarding the five aggregation vectors that are computed to be different than $\mathbf{e}$, $\mathbf{a}_{(0,1,0,0,1)}$, $\mathbf{a}_{(0,1,0,0,4)}$, $\mathbf{a}_{(0,1,2,2,4)}$, and $\mathbf{a}_{(0,1,3,3,4)}$ are used in aggregating diagonal blocks of $\tilde{Q}_{(0,0)}$, and their contributions are to the diagonals of diagonal blocks. The contributions of $\mathbf{a}_{(0,1,0,0,1)}$ and $\mathbf{a}_{(0,1,0,0,4)}$ in the first summation of $\tilde{Q}_{0,1}(0,0)$ are to the diagonal since $Q_1^{(2)}(\mathcal{R}_0^{(2)}, \mathcal{R}_0^{(2)}) = Q_4^{(2)}(\mathcal{R}_0^{(2)}, \mathcal{R}_0^{(2)}) = I_1$ and $Q_1^{(3)}(\mathcal{R}_0^{(3)}, \mathcal{R}_0^{(3)}) = Q_4^{(3)}(\mathcal{R}_0^{(3)}, \mathcal{R}_0^{(3)}) = I_2$. But their effects are canceled by the second negated summation of $\tilde{Q}_{0,1}(0,0)$ simply because $\mathrm{diag}(Q_1^{(2)}(\mathcal{R}_0^{(2)}, \mathcal{R}_0^{(2)})\mathbf{e}) = \mathrm{diag}(Q_4^{(2)}(\mathcal{R}_0^{(2)}, \mathcal{R}_0^{(2)})\mathbf{e}) = I_1$ and $\mathrm{diag}(Q_1^{(3)}(\mathcal{R}_0^{(3)}, \mathcal{R}_0^{(3)})\mathbf{e}) = \mathrm{diag}(Q_4^{(3)}(\mathcal{R}_0^{(3)}, \mathcal{R}_0^{(3)})\mathbf{e}) = I_2$. This is also the case for $\mathbf{a}_{(0,1,2,2,4)}$ and $\mathbf{a}_{(0,1,3,3,4)}$. Hence, we may very well set these four aggregation vectors to $\mathbf{e}$ as suggested before. Therefore, from $\tilde{Q}_{(0,1)} = P_{\mathbf{x}_{(0,0)}} \tilde{Q}_{(0,0)} R_{(0)}$, we implicitly have

$$\tilde{Q}_{(0,1)} = \begin{array}{c} \\ 0\ 0 \\ 0\ 1 \\ 0\ 0 \\ 0\ 1 \\ 1\ 0 \\ 1\ 1 \\ 0\ 2 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 2 \end{array} \\ \left(\begin{array}{cc|cc|cc|c} -\frac{280}{33} & 6 & \frac{16}{33} & & 2 & & \\ 6 & -\frac{448}{39} & & \frac{19}{39} & & 2 & 3 \\ \hline 2 & & -2 & & & & \\ & 2 & 6 & -8 & & & \\ \hline 4 & & & & -4 & & \\ & 4 & & & 6 & -10 & \\ \hline & 6 & & & & & -6 \end{array}\right) \end{array} .$$

In the next step, similar operations will be carried out at level 1 unless the aggregated generator matrix is solved exactly, upon which backtracking from recursion starts for iteration *it*.

The ML method we discussed follows a *V–cycle* [205, 206] at each iteration. That is, starting from the finest level, at each step, it smooths the current approximation and moves to a coarser level by aggregation until it reaches a level at which the aggregated generator matrix can be solved exactly. Once the exact solution is obtained at the coarsest level, the method starts moving in the opposite direction. At each step on the way to the finest level, the method disaggregates the current approximation passed by the coarser level and smooths it. Furthermore, the submodel state spaces, $\mathcal{S}^{(h)}$, are aggregated according to the *fixed* order $h = 1, \ldots, H$. However, to the contrary of the ML method for sparse MCs in [187], the definition of the aggregated state spaces follows naturally from the Kronecker representation in (2.10), and the aggregated generator matrices can also be represented using Kronecker products as shown in (6.23).

In [59], a sophisticated class of ML methods are given. The methods therein are capable of using JOR and SOR as smoothers, performing the *W–* and *F–cycles* inspired by *multigrid* [39, 276, 324] and aggregating submodel state spaces according to *circular* and *adaptive* orders. Here, a *cycle* may be viewed as the operations corresponding to an outer ML iteration denoted by *it* in (6.16), and W–cycle refers to invoking at each level two recursive calls to the next coarser level, whereas an F–cycle at a level can be viewed as a recursive call to a W–cycle followed by a recursive call to a V–cycle at the next coarser level. In the circular order of aggregation, at the beginning of each ML cycle at the finest level, a circular shift of submodel indices is performed to achieve fairness in aggregating submodel state spaces. Hence, every H cycles each submodel will have received the opportunity to get aggregated first. On the other hand, in the adaptive order of aggregation, submodel indices are sorted according to the residual norms restricted to the corresponding submodel state space at the end of the ML cycle, and aggregation of submodels in this sorted order in the next cycle is performed. This ensures that submodels which have smaller residual norms are aggregated earlier at finer levels, since small residual norms are expected to be indicative of good numerical results in those submodels. ML methods discussed in this section are coded within the `NSolve` package of the APNN toolbox [7, 22]. Numerical experiments in [59, 61] prove these methods to be very strong, robust, and scalable solvers for MCs based on Kronecker products.

The convergence properties of the class of ML methods in [59] are discussed in [61]. An almost positive row or column in the iteration matrices of the smoother with $\eta_{(it,l)}$ pre- and $\nu_{(it,l)}$ post-smoothings at each level l across iterations it is shown to facilitate convergence locally, and SOR is to be recommended among the three smoothers considered. An error propagation formula for ML methods in which the number of pre-smoothings is set to zero is introduced in [272]. Therein are examples indicating that a two-level method may be converging while the corresponding three-level method is not and vice versa. One important question to be answered is whether local convergence implies global convergence (even at two levels). Furthermore, it

is not clear how the behavior of the ML method would be affected if block iterative methods, such as BJOR and BSOR, are used as smoothers rather than power, JOR, and SOR. Note that BJOR and BSOR should normally not use a direct method for the solution of the diagonal blocks when employed as smoothers within the ML method, since the aggregated generator matrix at each level changes from iteration to iteration and the LU factorization of the diagonal blocks may be too time consuming to offset.

In [173] an efficient algorithm that finds an NCD [97, 123, 241, 305] partitioning of $\mathcal{S}$ in the presence of functional transitions for a user-specified decomposability parameter is given. Since IAD using NCD partitionings has certain rate of convergence guarantees [304] (cf. [225, 228, 271]), the algorithm may be useful in the context of ML methods to determine the loosely coupled dimensions to be aggregated first in a given iteration.

Distributed implementation of block iterative methods coupled with some number of aggregation–disaggregation steps for HMMs is investigated on a cluster of workstations in [67]. Therein, especially asynchronous computation models that observe coarse-grained parallelism (meaning large chunks of computations interleaved with small amount of communications) based on the nested block partitioning of Q are recommended at lower level numbers. Along a different line, the performance of a prototype parallel version of PEPS on a cluster architecture is modeled using SANs and analyzed theoretically with a sequential version of PEPS [14]. The study in [288] concentrates on the parallelization of the vector operations of initialization, reciprocation, aggregation, disaggregation, scaling, addition, multiplication and the matrix operations of vector–matrix multiplication and aggregation in symbolic ML [287] with V–cycle, fixed order of submodel aggregation, and JOR smoother under the *multiterminal binary decision diagram* (MTBDD) data structure using multiple threads. Clearly, there are issues that need to be addressed further such as the level at which parallelism is employed, resolving data collision due to multiple threads accessing data that contributes to the same intermediate result and load balancing across threads. The results however are quite encouraging and indicate that there is ample opportunity for parallelism in the implementation of iterative methods based on the block partitioning of Q and the sum of Kronecker products forming each nonzero block.

6.4 Decompositional Methods

Among the iterative methods discussed in the previous sections, ML methods perform better on a larger number of problems in the literature [59, 61]. However, there are certain classes of problems for which other methods could be preferred. One such method we present in this section is iterative and based on *decomposing* a model into its submodels, analyzing the submodels individually for their steady-state, and putting back the individual solutions

together using disaggregation in a correction step [15]. This method is able to compute the steady-state solution exactly up to computer precision to a model having weakly interacting submodels in a relatively small number of iterations and with modest memory requirements.

Consider the separation of Q_O in (6.2) into two terms as in

$$Q_O = Q_{local} + Q_{synch} \ ,$$

where Q_{local} and Q_{synch} correspond to those parts of Q_O associated with local and synchronizing transitions, respectively. Without loss of generality, we adopt the enumeration of the K terms as in Example 1 and let the first H represent local transitions, the hth being associated with submodel h for $h = 1, \ldots, H$. The remaining $(K - H)$ terms correspond to synchronizing transitions. Hence,

$$Q_{local} = \sum_{k=1}^{H} \bigotimes_{h=1}^{H} Q_k^{(h)} \quad \text{and} \quad Q_{synch} = \sum_{k=H+1}^{K} \bigotimes_{h=1}^{H} Q_k^{(h)} \ .$$

Recall that this enumeration necessarily implies $Q_k^{(h)} = I_{n_h}$ for $k \neq h$ and $h = 1, \ldots, H$ due to the definition of local transitions. In Example 1, we have $\mathcal{S} = \mathcal{R}$; hence, there is a single reachable state space partition for which $|\mathcal{R}| = \prod_{h=1}^{H} |\mathcal{S}^{(h)}| = \prod_{h=1}^{H} |\mathcal{R}^{(h)}| = \prod_{h=1}^{H} n_h = n$. Furthermore, we observe that the irreducibility of Q does not imply the irreducibility of the local transition rate matrices $Q_h^{(h)} \in \mathbb{R}_{\geqslant 0}^{n_h \times n_h}$ for $h = 1, \ldots, H$.

Now, let $Q = U - L$ be the forward GS splitting of the generator matrix in Kronecker form, where $U = Q_D + Q_{U(H)}$ corresponds to its upper-triangular part and $L = Q_{L(H)}$ contains its negated strictly lower-triangular part as in (6.8). Furthermore, let the aggregation operator $R^{(h)} \in \mathbb{R}_{\geqslant 0}^{n \times n_h}$ (cf. (6.18)) for $h = 1, \ldots, H$ be associated with the mapping $f_{(h)} : \mathcal{R} \mapsto \mathcal{R}^{(h)}$ and have its $(\mathbf{i}, i_h)$th entry be given by

$$r^{(h)}(\mathbf{i}, i_h) = \begin{cases} 1 & \text{if } f_{(h)}(\mathbf{i}) = i_h \\ 0 & \text{otherwise} \end{cases} \quad \text{for } \mathbf{i} \in \mathcal{R} \text{ and } i_h \in \mathcal{R}^{(h)} \ .$$

Observe that the mapping $f_{(h)}$ represents the aggregation of all dimensions except the hth. In Kronecker form,

$$R^{(h)} = \left(\bigotimes_{l=1}^{h-1} I_{n_l} \mathbf{e} \right) \otimes I_{n_h} \otimes \left(\bigotimes_{l=h+1}^{H} I_{n_l} \mathbf{e} \right) \quad \text{for } h = 1, \ldots, H \ . \tag{6.24}$$

On the other hand, let the disaggregation operator $S_{\boldsymbol{\pi}_{(it)}}^{(h)} \in \mathbb{R}_{\geqslant 0}^{n_h \times n}$ (cf. (6.19)) for $h = 1, \ldots, H$ be associated with the mapping $f_{(h)}$ and have its $(i_h, \mathbf{i})$th entry be given by

$$s^{(h)}_{\boldsymbol{\pi}_{(it)}}(i_h, \mathbf{i}) = \begin{cases} \frac{\boldsymbol{\pi}_{(it)}(\mathbf{i})}{\boldsymbol{\pi}^{(h)}_{(it)}(i_h)} & \text{if } f_{(h)}(\mathbf{i}) = i_h \\ 0 & \text{otherwise} \end{cases} \quad \text{for } \mathbf{i} \in \mathcal{R} \text{ and } i_h \in \mathcal{R}^{(h)} ,$$

where

$$\boldsymbol{\pi}^{(h)}_{(it)} := \boldsymbol{\pi}_{(it)} R^{(h)}.$$

Then the decompositional iterative method can be stated [15] for a user-specified `stop_tol` as in Algorithm 8.

ALGORITHM 8. *Decompositonal iterative method with GS correction step.*

$it := 0$; $\mathbf{y}_{(it)} := \mathbf{0}$; $\boldsymbol{\pi}_{(it)} := \mathbf{e}^T/n$;
Repeat
  For $h := 1$ to H,
    If $Q^{(h)}_h$ is irreducible, solve $\boldsymbol{\pi}^{(h)}_{(it+1)} Q^{(h)}_h = \mathbf{v}^{(k)}(\boldsymbol{\pi}_{(it)})$,
      where $\mathbf{v}^{(k)}(\boldsymbol{\pi}_{(it)}) := -\boldsymbol{\pi}_{(it)} Q_{synch} R^{(h)}$,
    Else solve $\boldsymbol{\pi}^{(h)}_{(it+1)} Q^{(h)}(\boldsymbol{\pi}_{(it)}) = \mathbf{0}$,
      where $Q^{(h)}(\boldsymbol{\pi}_{(it)}) := Q^{(h)}_h + S^{(h)}_{\boldsymbol{\pi}_{(it)}} Q_{synch} R^{(h)}$,
    subject to $\boldsymbol{\pi}^{(h)}_{(it+1)} \mathbf{e} = 1$;
  Solve $\mathbf{y}_{(it+1)} U = \mathbf{y}_{(it)} L + \left(\bigotimes_{h=1}^{H} \boldsymbol{\pi}^{(h)}_{(it+1)} \right) Q$;
  $\boldsymbol{\pi}_{(it+1)} := \left(\bigotimes_{h=1}^{H} \boldsymbol{\pi}^{(h)}_{(it+1)} \right) - \mathbf{y}_{(it+1)}$ subject to $\boldsymbol{\pi}_{(it+1)} \mathbf{e} = 1$;
  $it := it + 1$;
Until $\|\boldsymbol{\pi}_{(it)} Q\|_\infty <$ `stop_tol`.

The algorithm starts by initializing the correction vector, $\mathbf{y}_{(it)}$, to zero and the solution vector, $\boldsymbol{\pi}_{(it)}$, to the uniform distribution, respectively. Then each system of local equations is solved subject to a normalization condition. If $Q^{(h)}_h$ is irreducible, then a unique new local solution vector $\boldsymbol{\pi}^{(h)}_{(it+1)} \in \mathbb{R}^{1 \times n_h}_{>0}$ can be computed. This is so, because each system to be solved has a zero sum right-hand side vector, $\mathbf{v}^{(h)}(\boldsymbol{\pi}_{(it)})$, (i.e., $\mathbf{v}^{(h)}(\boldsymbol{\pi}_{(it)})\mathbf{e} = 0$) due to the particular way in which synchronizing transition rate matrices, $Q^{(h)}_k \in \mathbb{R}^{n_h \times n_h}_{\geqslant 0}$ for $k = H+1, \ldots, K$, are specified. On the other hand, when $Q^{(h)}_h$ is reducible, we consider a homogeneous system in which the aggregated matrix $Q^{(h)}(\boldsymbol{\pi}_{(it)})$ is used. The aggregated matrix is irreducible if Q is irreducible and $\boldsymbol{\pi}_{(it)} \in \mathbb{R}^{1 \times n}_{>0}$. Hence, the existence of a unique $\boldsymbol{\pi}^{(h)}_{(it+1)} \in \mathbb{R}^{1 \times n_h}_{>0}$ is also guaranteed in this case. Since $Q^{(h)}_k$ for $k = H+1, \ldots, K$ are in general very sparse, the enumeration process associated with the nonzeros in Q_{synch} to form $\mathbf{v}^{(k)}(\boldsymbol{\pi}_{(it)})$, or $Q^{(h)}(\boldsymbol{\pi}_{(it)})$, can be handled systematically. Note that there are differences from a computational point of view between the two alternative solution steps. In the former case, $Q^{(h)}_h$ is constant and already available in row sparse format; the right-hand side vector is dependent on

$\boldsymbol{\pi}_{(it)}$. In the latter case, $Q^{(h)}(\boldsymbol{\pi}_{(it)})$ needs to be reconstructed at each iteration, and it is the coefficient matrix that is dependent on $\boldsymbol{\pi}_{(it)}$. Hence, the two approaches for obtaining $\boldsymbol{\pi}^{(h)}_{(it+1)}$ are not equivalent except at steady-state (i.e., $\boldsymbol{\pi}_{(it)} = \boldsymbol{\pi}_{(it+1)} = \boldsymbol{\pi}$).

In the next step, the new correction vector, $\mathbf{y}_{(it+1)}$, is obtained by solving an upper-triangular system in Kronecker form. Since Q is assumed to be irreducible, $\mathbf{y}_{(it+1)}$ is computed through a GS relaxation on Q with a zero sum but nonzero right-hand side.

The last step subtracts $\mathbf{y}_{(it+1)}$ from the Kronecker product of $\boldsymbol{\pi}^{(h)}_{(it+1)}$ for $h = 1, \ldots, H$ to form the new solution vector, $\boldsymbol{\pi}_{(it+1)}$, and then the iteration number is incremented. These steps are repeated until the residual infinity norm becomes smaller than `stop_tol`.

Algorithm 8 is coded within the `NSolve` package of the APNN toolbox [7, 22] and numerical experiments are carried out [15] on larger and slightly different versions of Example 1. The decompositional method is compared with point iterative methods based on splittings, BGS, projection methods, BGS preconditioned projection methods, and ML with one pre- and one post-smoothing using GS, W–cycle, and circular order of aggregating submodels in each cycle. The solvers are compared in terms of number of iterations to converge to `stop_tol` $:= 10^{-8}$ on the infinity norm of the residual, elapsed CPU time, and amount of allocated main memory. The diagonal blocks associated with the BGS solver and the BGS preconditioner for projection methods at a suitable level number are LU factorized [58] using `colamd` ordering [94, 95]. The number of nonzeros generated during the LU factorization of the diagonal blocks is accounted for in the memory consumed by solvers utilizing BGS.

It is observed that convergence becomes very fast for the decompositional solver when the synchronizing transition rates are small since the submodels in that case are nearly independent and the Kronecker product of the local solutions yields a very good approximation to the solution early in the iteration. The solver corresponding to Algorithm 8 requires modest memory and a small number of iterations for relatively fast convergence to the solution. The second best solver is ML, which trails in all three areas. It improves only slightly as the synchronizing transition rates become smaller. Projection methods do not benefit from BGS preconditioning. The performances of the point iterative methods based on splittings and BGS are not affected by a change in the rates of synchronizing transitions. BGS performs very poorly due to the large time per iteration. BGS and BGS preconditioned projection methods require considerably more memory than the other methods, because of the need to store factors of diagonal blocks and, in the latter case, also a larger number of vectors. Memorywise, the decompositional solver requires about 1.5 times that of point iterative methods based on splittings, but less than ML, and therefore can be considered to be memory efficient.

The scalability of the decompositional solver is investigated for increasing number of submodels when the synchronizing transition rates are relatively small compared to those in local transition rate matrices. It is observed that the number of iterations to converge decreases as the number of submodels increases. This is due to the decrease in the throughputs of synchronizing transitions for a larger number of submodels (because the steady-state probabilities of states in which synchronizing transitions can take place become smaller), leading to more independent submodels. This is different from the behavior of the ML method, which takes more or less the same number of iterations to converge as the number of submodels increases. The scalability of the decompositional solver is also investigated for increasing number of synchronizing transitions when the number of submodels is kept constant and the rates of synchronizing transitions are relatively small compared to those in the local transition rate matrices. As expected, the results indicate that the time the decompositional solver takes to converge is affected linearly by an increase in the number of synchronizing transitions.

There are also iterative methods based on *polyhedra* theory [86] and disaggregation, such as the one in [55] for SANs which provides satisfactory lower and upper *bounds* on the solution only if the interactions among submodels are weak or the rates of synchronizing transitions are more or less independent of the states of submodels. Another class of iterative methods are those that are *approximative*. For instance, the method in [56] for superposed GSPNs operates at a fine level only on states having higher steady-state probabilities; the remaining states are aggregated and treated at a coarse level. The steady-state vector can be stored with significant savings due to its compact representation as a Kronecker product of the aggregated submodel steady-state vectors. An approximative class of iterative methods are also presented in [101]. The approximative methods therein are geared toward *closed* networks of first-come first-served (FCFS) queues with PH service distributions and arbitrary buffer sizes when a few digits of accuracy in the computed solution are sufficient for analysis purposes [104]. The analysis of closed queueing networks with PH service distributions and arbitrary buffer sizes is challenging due to the fact that the corresponding state spaces grow exponentially with numbers of customers, queues, and phases in the service distribution of each queue. Now, we discuss these approximative iterative methods for closed queueing networks, which are also based on decomposition. More information regarding this work is available in [104, 239].

Queueing networks have been used in the literature to model and analyze a variety of systems involving customers, packets, or jobs waiting to get service [180, 181, 306]. The work in [104] concentrates on a relatively large class of problems which do not possess analytical solutions [153]. Closedness implies that the number of customers circulating in the queueing network remains constant; there are no arrivals to the network from the outside, there are no departures to the outside, and the number of customers inside the network neither increases nor decreases as a result of the queueing discipline and the

service process. A customer departs from a queue after getting service and joins a(nother) queue, possibly the same one it departed from. If a queueing network is not closed, it is said to be *open*. Regarding service distributions, *hypoexponential*, *hyperexponential*, *Coxian*, and *Erlang* are all phase-type and have rational Laplace transforms. Furthermore, the exponential distribution is a special case of the Erlang distribution, which is yet a special case of the hypoexponential distribution. Interestingly, it is proved that Erlang is the most suitable phase approximation for the *deterministic* distribution [6]. This is taken advantage of when modeling a robotic tape library [119] and a multiprocessor system [285] using SANs. For practical purposes, a five- to ten-phase Erlang is considered sufficient for approximating a deterministic distribution. The use of PH distributions in SANs is further investigated in [284].

In [104], two approximative decompositional iterative methods are addressed, the first of which appears in [235] and the second one in [329]. Each decomposes the network into subnetworks. They differ in the way the decompositions are obtained, and the solutions to subnetworks are put together. These approximative methods require the modeling of subnetworks whose product state space sizes are larger than their reachable state space sizes and are shown that they can be implemented using Kronecker products.

In its setup phase, the first method in [235] partitions the closed queueing network into subnetworks. In doing this, it classifies each queue as finite or infinite buffer. Finite buffer queues are those that have positive blocking probabilities for a given total number of customers circulating in the closed queueing network. The method places queues feeding a finite buffer queue in the same subnetwork with the finite buffer queue. In this way, the method aims at achieving a decomposition in which transition probabilities between subnetworks are independent of the states of the subnetworks. Thus, each subnetwork can be considered as a service station with state-dependent exponential service rate for which the parameters of the equivalent server are obtained by analyzing the subnetwork in isolation as an open queueing network assuming it has state-dependent arrivals with exponentially distributed interarrival times. This is done by modeling the open queueing network as a closed queueing network, which consists of the subnetwork's queues and a slack queue as in Figure 6.1. The slack queue is assumed to have a finite buffer of size equal to the total number of customers circulating in the closed queueing network formed of multiple subnetworks. The state-dependent throughputs are the state-dependent service rates since the slack queue is practically infinite. Each closed queueing network obtained as such can be modeled by defining the queues in the subnetwork and the slack queue, and then constructing a block matrix which represents the interactions among the queues in the closed queueing network using Kronecker products as in [43]. The approximate results are obtained via *fixed-point iteration*, which requires throughputs of subnetworks to be computed. For this purpose, an analytical method, the convolution algorithm (CA) [160] is used. We remark that

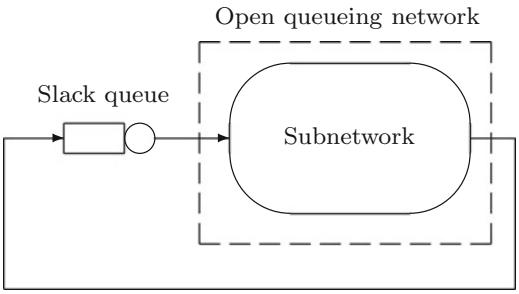


Fig. 6.1 Open queueing network corresponding to subnetwork modeled as closed queueing network with slack queue having finite buffer.

a fixed-point iteration can be perceived, for instance, as an iteration of the form in block iterative methods, for which the solution, π, that is sought is the *fixed-point* of the system of equations, $\pi = \pi T$, to be solved [305].

On the other hand, the second decompositional method in [329] partitions the closed queueing network into individual queues and approximates the service distribution of each queue by a state-dependent exponential service distribution. Thus, the method transforms the closed queueing network into another closed queueing network with state-dependent exponential service distributions. The decomposition in this approach is maximal, meaning each queue is placed in a separate subnetwork. Again a slack queue with an infinite buffer and a state-dependent exponential service distribution is used to model state-dependent arrivals with exponentially distributed interarrival times to each queue having a PH service distribution. After this approximation, the method sets the state-dependent service rate of the slack queue to some initial value and then employs a fixed-point iteration on the decomposed network to compute the throughputs of all queues. Again, initialization of the state-dependent service rates of slack queues, which are their state-dependent throughputs, can be done using CA.

Implementations of the two approximative decompositional methods (respectively, named M and YB hereafter) are available in Matlab [240] together with implementations of CA, a *mean value analysis algorithm for blocking closed queueing networks* (MVABLO) [5], point iterative methods based on splittings, and ML with fixed and circular orders of aggregation in V–, F–, and W–cycles. Experiments are performed to compare the methods for their accuracy and efficiency on various models for analyzing utilizations and mean lengths of queues. ML and GS methods assume `stop_tol` $:= 10^{-15}$ on the residual 1-norm. ML uses GS as the smoother and performs one pre- and one post-smoothing at each level. A `stop_tol` $:= 10^{-4}$ is used on the approximate error of utilizations and mean lengths of queues for M and YB. The subnetworks resulting from decomposition in these methods are solved with ML. When computing the steady-state vector of the coarsest generator matrix in M and the steady-state vector of the state-dependent closed

queueing network with exponential service distributions in YB, if the order of the matrix is less than 500, GE, otherwise BICGSTAB with ILU preconditioning and a drop tolerance of 10^{-5} [123] is employed. Results obtained by approximative methods are compared with results of ML and , relative errors are provided using the 1-norm. Note that relative errors are indicative of numbers of correct decimal digits in the results. That is, an approximate result with a relative error in the order of 10^{-z} implies z correct decimal digits.

CA and MVABLO produce acceptable results for problems with balanced service requirements and relatively small number of blocking queues. On the other hand, M and YB provide relatively more accurate results for all problems and yield results with at least 2 digits of accuracy for unbalanced service requirements. Also, unlike the results obtained with CA and MVABLO, an increase in the number of blocking queues has almost no effect in the results obtained with M and YB. Therefore, M and YB emerge as more accurate methods than CA and MVABLO for problems with unbalanced service requirements and relatively large number of blocking queues. When accuracies of M and YB are compared, especially in problems with unbalanced service requirements and relatively large number of blocking queues, M can produce at least half a digit more accurate results for utilizations than YB.

When efficiencies of M and YB are compared, it can be seen that the number of flops performed by YB to compute arrival rates of queues mostly depends on the number of flops performed for obtaining the solution of the state-dependent closed queueing network with exponential service distributions generated at each fixed-point iteration. Therefore, for problems which require a relatively small number of flops for the solution of this queueing network, YB executes less flops than M. Also, for problems which result in subnetworks with a relatively large number of queues for M, YB may end up performing less flops than M through its fixed-point approximation process. Consequently, efficiencies of M and YB depend heavily on the particular problem. Nevertheless, the average number of fixed-point iterations performed by M and YB over all problems considered are 4 and 5, respectively. When ML and GS are compared, we see that ML achieves convergence within 100 iterations in all problems. On the other hand, GS does not converge within a reasonable number of iterations or time in some of the problems. Clearly, the number of iterations determines the number of flops executed by the methods, and ML performs less flops than GS in almost all problems. Even though GS takes less space in memory than ML, in most of the problems, ML requires less memory than the sparse representation of the generator matrix underlying the closed queueing network with PH service distributions and arbitrary buffer sizes, thereby being capable of solving variants of the problems with relatively large numbers of customers. Since M and YB are based on decomposition, the space requirements of M and YB are smaller than those of ML and GS for relatively large problems. Indeed, it is verified that the usage of ML in M and YB introduces another dimension of scalability to the space requirements of the two methods.

6.5 Matrix Analytic Methods

Continuous-time LDQBDs are CTMCs having generator matrices that can be symmetrically permuted to the block tridiagonal form

$$Q = \begin{pmatrix} Q(0,0) & Q(0,1) & & & & \\ Q(1,0) & Q(1,1) & Q(1,2) & & & \\ & \ddots & \ddots & \ddots & & \\ & & Q(l,l-1) & Q(l,l) & Q(l,l+1) & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{6.25}$$

As opposed to QBDs [4, 33, 213], the dependency on level number, l, in (6.25) manifests itself with two indices rather than one in each nonzero block for $l \in \mathbb{Z}_{\geqslant 0}$. It is the first index which corresponds to level number. Nevertheless, nonzero blocks can be dependent on level number in two different ways. It is either the nonzero values or the dimensions of a nonzero block (or both) that depend on level number. In this respect, LDQBDs generalize QBDs with nonhomogeneous transition rates and rectangular subdiagonal/superdiagonal nonzero blocks.

Assuming that the subset of states corresponding to level l is denoted by $\mathcal{R}_l$, the nonzero blocks at level l are given by

$$Q(l,l-1) \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}_l| \times |\mathcal{R}_{l-1}|}, \quad Q(l,l) \in \mathbb{R}^{|\mathcal{R}_l| \times |\mathcal{R}_l|}, \quad Q(l,l+1) \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}_l| \times |\mathcal{R}_{l+1}|}.$$

Negative entries appear only along the diagonal of $Q(l,l)$. There are a countably infinite number of levels, and transitions from level l are either to states within itself or to states in the adjacent levels $(l-1)$ and $(l+1)$. Level 0 is an exception since it constitutes the boundary level and has two nonzero blocks. Clearly, the ordering of states within a level is fixed only up to a permutation.

Assuming that steady-state exists, thereof the probability distribution vector may be written in a piecemeal manner as

$$\boldsymbol{\pi} = (\boldsymbol{\pi}(\mathcal{R}_0), \boldsymbol{\pi}(\mathcal{R}_1), \ldots),$$

and its subvector at level $(l+1)$ can be obtained from

$$\boldsymbol{\pi}(\mathcal{R}_{l+1}) = \boldsymbol{\pi}(\mathcal{R}_l) R_l \tag{6.26}$$

once the *matrix of conditional expected sojourn times* at level l

$$R_l = Q(l,l+1)(-Q(l+1,l+1) - R_{l+1}Q(l+2,l+1))^{-1} \tag{6.27}$$

is available for $l \in \mathbb{Z}_{\geqslant 0}$ [40]. In (6.27), $R_l(\mathbf{i},\mathbf{j})$ records the expected sojourn time in state $\mathbf{j} \in \mathcal{R}_{l+1}$ per unit sojourn in state $\mathbf{i} \in \mathcal{R}_l$ before returning to level l, given that the process started in state $\mathbf{i}$ [274]. We remark that

$$R_l \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}_l| \times |\mathcal{R}_{l+1}|} \quad \text{for } l \in \mathbb{Z}_{\geqslant 0}$$

is nonnegative and rectangular. The recurrence in (6.26) requires $\boldsymbol{\pi}(\mathcal{R}_0)$ to be determined first. This can be done from the set of boundary equations

$$\boldsymbol{\pi}(\mathcal{R}_0)Q(0,0) + \boldsymbol{\pi}(\mathcal{R}_1)Q(1,0) = \mathbf{0}$$

corresponding to the first column of blocks using $\boldsymbol{\pi}(\mathcal{R}_1) = \boldsymbol{\pi}(\mathcal{R}_0)R_0$ from s(6.27). Hence, we conclude that $\boldsymbol{\pi}(\mathcal{R}_0)$ should be the positive left eigenvector of $Q(0,0) + R_0Q(1,0)$ corresponding to the eigenvalue 0 [242]. Eventually, $\boldsymbol{\pi}$ should be normalized so that $\boldsymbol{\pi}\mathbf{e} = 1$.

In Section 4.4, we have discussed how the countable infiniteness of the reachable state space

$$\mathcal{R} = \bigcup_{p=0}^{\infty} \mathcal{R}_p$$

in (4.5) can be handled during steady-state analysis when Q is ergodic using a suitable Lyapunov function [157, 314]. We have also discussed how states are assigned to reachable state space partitions, $\mathcal{R}_p$, and had given a Kronecker representation for each nonzero block. In fact, reachable state space partitions as defined in the metabolite synthesis and call center models correspond to levels in LDQBDs, that is, $\mathcal{R}_p = \mathcal{R}_l$ for $p = l$ and $p, l \in \mathbb{Z}_{\geqslant 0}$, since Q in each of these models for the given partitioning of the reachable state space $\mathcal{R}$ is block tridiagonal. It should be apparent from the values $\mathbf{i} \in \mathcal{R}$ can take that the state space $\mathcal{R}$ is countably infinite and an LDQBD model requires us to truncate it judiciously for analysis purposes.

In many cases, an LDQBD can be shown to be ergodic by checking easy to verify conditions on the *birth-and-death process* (i.e., a one-dimensional CTMC in which transitions from state i to states $i-1$, a death, and $i+1$, a birth, is possible) defined over its levels [268]. However, for computational purposes, we prefer to consider Lyapunov functions, since it is through this approach that lower and higher level numbers (called Low and $High$, respectively) can be computed [103, 107, 120] between which a specified percentage of the steady-state probability mass lies when the LDQBD is ergodic. Once we have proved the finiteness of $\mathcal{C}$ and determined χ (or equivalently, γ for chosen ε) with the help of a suitable Lyapunov function using (4.3), we can compute the pair of level numbers, $(Low, High)$, of the LDQBD such that the states within levels Low to $High$ include all the states in $\mathcal{C}$. In other words, we set

$$Low = \min\{l \in \mathbb{Z}_{\geqslant 0} \mid \mathcal{R}_l \cap \mathcal{C} \neq \varnothing\} \quad \text{and} \quad High = \max\{l \in \mathbb{Z}_{\geqslant 0} \mid \mathcal{R}_l \cap \mathcal{C} \neq \varnothing\}\,,$$

and the finite set

$$\tilde{\mathcal{R}} := \bigcup_{l=Low}^{High} \mathcal{R}_l$$

contains at least $1-\varepsilon$ of the steady-state probability. We remark that $\mathcal{C} \subseteq \tilde{R}$ due to the way in which Low and $High$ are defined, but using a truncated set of states which is a superset of $\mathcal{C}$ can only improve the quality of the bound.

Given the pair of level numbers, $(Low, High)$, the next step is to compute the R_l matrices of conditional expected sojourn times for levels between Low and $High$ using (6.27). This requires us to determine a starting value for R_{High}. Since R_{High} can only be approximated when the state space is truncated, all computed R_l matrices of conditional expected sojourn times between levels Low and $High$ will also be approximate. Clearly, the quality of the approximations improves as the steady-state probability mass concentrated on states within levels Low and $High$ approaches 1 (i.e., as ε approaches 0) [120]. Through a set of experiments in [120], it is shown that setting R_{High} to 0 as suggested in [20] results in almost no loss of accuracy when ε is close to 0 and is the best overall choice.

The matrix analytic computation of the steady-state vector of an LDQBD is given in Algorithm 9 [19]. Its implementation within the `NSolve` package of the APNN toolbox [7, 22] is available at [109]. At step l, the linear system of equations

$$\tilde{R}_l A_l = B_l \quad \text{for } l = High - 1 \text{ down to } Low$$

is solved for the rectangular matrix $\tilde{R}_l \in \mathbb{R}_{\geqslant 0}^{|\mathcal{R}_l| \times |\mathcal{R}_{l+1}|}$ of unknowns, where

$$A_l := Q(l+1, l+1) + \tilde{R}_{l+1} Q(l+2, l+1)$$

is the square coefficient matrix such that $A_l \in \mathbb{R}^{|\mathcal{R}_{l+1}| \times |\mathcal{R}_{l+1}|}$ and

$$B_l := -Q(l, l+1)$$

is the rectangular matrix of multiple right-hand sides with $B_l \in \mathbb{R}^{|\mathcal{R}_l| \times |\mathcal{R}^{l+1}|}$. Clearly, $\tilde{R}_{l+1}$ must be available at step l for the computation to proceed. That R_l and $\pi(\mathcal{R}_l)$ become approximations once R_{High} is set to 0 is indicated by using them with tilde.

The computation of $\tilde{R}_l$ requires the nonzero blocks $Q(l+1, l+1)$ and $Q(l+2, l+1)$ to be obtained, A_l to be formed by multiplying $\tilde{R}_{l+1}$ with $Q(l+2, l+1)$, and then $Q(l+1, l+1)$ to be added to the product. At the end, A_l should be LU factorized and the linear system solved for each right-hand side vector in B_l. The matrix–matrix multiplication $\tilde{R}_{l+1} Q(l+2, l+1)$ can be handled in two different ways [18, 19]. First, $Q(l+2, l+1)$ may be generated from the Kronecker representation as a sparse matrix and the pre-multiplication with $\tilde{R}_{l+1}$ performed. Second, the efficient vector–Kronecker product algorithm [101, 136] can be used to multiply rows of $\tilde{R}_{l+1}$ with the

ALGORITHM 9. *Matrix analytic computation of steady-state vector of an LDQBD.*

Choose a suitable Lyapunov function $g(\mathbf{i})$ proving ergodicity:
  Choose $g(\mathbf{i})$ such that the set of states for which $g(\mathbf{i}) < \infty$ is finite;
  Obtain the drift $d(\mathbf{i})$ and show that it is bounded;
  $\chi := \sup_{\mathbf{i}\in\mathcal{R}} d(\mathbf{i})$ (using HOM4PS2-2.0 if necessary);
  $\gamma := \chi(1/\varepsilon - 1)$ for given ε;
  Show that $\mathcal{C} = \{\mathbf{i} \in \mathcal{R} \mid d(\mathbf{i}) > -\gamma\}$ is finite;
Compute $(Low, High)$;
$\tilde{R}_{High} := 0$;
For $l := High - 1, \ldots, Low$,
  $A_l := Q(l+1, l+1) + \tilde{R}_{l+1}Q(l+2, l+1)$;
  $B_l := -Q(l, l+1)$;
  Solve $\tilde{R}_l A_l = B_l$ for $\tilde{R}_l$;
$A_{Low-1} := Q(Low, Low) + \tilde{R}_{Low}Q(Low+1, Low)$;
Solve $\tilde{\boldsymbol{\pi}}(\mathcal{R}_{Low})A_{Low-1} = \mathbf{0}$ for $\tilde{\boldsymbol{\pi}}(\mathcal{R}_{Low})$ subject to $\|\tilde{\boldsymbol{\pi}}(\mathcal{R}_{Low})\|_1 = 1$;
For $l := Low, \ldots, High - 1$,
  $\tilde{\boldsymbol{\pi}}(\mathcal{R}_{l+1}) := \tilde{\boldsymbol{\pi}}(\mathcal{R}_l)\tilde{R}_l$;
  Normalize $\tilde{\boldsymbol{\pi}}$ subject to $\|(\tilde{\boldsymbol{\pi}}(\mathcal{R}_{Low}), \ldots, \tilde{\boldsymbol{\pi}}(\mathcal{R}_{l+1})\|_1 = 1$;
Compute $\|\boldsymbol{\pi}Q\|_1$ by letting
  $\boldsymbol{\pi} := (\mathbf{0}(\mathcal{R}_0)^T, \ldots, \mathbf{0}(\mathcal{R}_{Low-1})^T, \tilde{\boldsymbol{\pi}}(\tilde{\mathcal{R}}), \mathbf{0}(\mathcal{R}_{High+1})^T, \ldots)$.

block $Q(l+2, l+1)$, while the latter operand is being held in Kronecker form. 1-norm of the residual vector of the countably infinite model [112] is computed from

$$\|\boldsymbol{\pi}Q\|_1 := \sum_{\mathbf{i}\in\tilde{\mathcal{R}}} \left|\tilde{\mathbf{r}}(\mathbf{i}) - \sum_{\mathbf{j}\notin\tilde{\mathcal{R}}} \tilde{\boldsymbol{\pi}}(\mathbf{i})Q(\mathbf{i},\mathbf{j})\right| + \sum_{\mathbf{i}\in\tilde{\mathcal{R}}}\sum_{\mathbf{j}\notin\tilde{\mathcal{R}}} \tilde{\boldsymbol{\pi}}(\mathbf{i})Q(\mathbf{i},\mathbf{j}) \quad \text{with} \quad \tilde{\mathbf{r}} := \tilde{\boldsymbol{\pi}}\tilde{Q},$$

where $\tilde{Q}$ is the truncated generator matrix. Once $(Low, High)$ is determined, computation of the steady-state vector of the LDQBD can also be performed using (block) GE [125, 158, 302, 305]. This approach can be equally efficient [166], but unfortunately does not provide the $\tilde{R}_l$ matrices.

Having observed in [107, 120] that the $\tilde{R}_l$ matrices are not necessarily sparse, their full and sparse storages are considered. When the $\tilde{R}_l$ matrices are stored as full matrices, a temporary matrix in full storage needs to be set aside to form A_l and then compute its LU factorization in place. Since $\tilde{R}_{High} := 0$, the sparsity pattern of A_{High-1} is equal to that of $Q(High, High)$. Therefore, $\tilde{R}_{High-1}$ should be obtained using sparse LU factorization even though it is later stored as a full matrix. Now, although all $\tilde{R}_l$ matrices are to be kept until the computation ends to obtain the steady-state solution and the sizes of A_l and $\tilde{R}_l$ matrices are known a priori at each level, it is still possible to consider two different memory allocation–deallocation schemes for these two matrices [18, 19]. First, memory to store $\tilde{R}_l$ matrices from $l = High - 1$ down to Low

and the largest temporary matrix, A_{High-1}, can be allocated at the outset and deallocated at the end of the computation. In this scheme, successive A_l matrices overwrite the same temporary matrix. Second, the memory of A_l can be deallocated at the end of step l, and the memory for $\tilde{R}_{l-1}$ and A_{l-1} can be allocated at the beginning of step $(l-1)$. When the temporary matrix is allocated once at the outset, peak total memory usage turns out to be higher especially when H_I, number of countably infinite dimensions, is larger. On the other hand, when $\tilde{R}_l$ matrices are chosen to be stored as sparse matrices, the temporary matrix to form A_l is also stored as a sparse matrix, and memory is allocated at each level. This requires the allocation of extra memory for the sparse LU factorization of A_l due to expected fill-in.

In the proposed Kronecker representations in Chapter 4 for countably infinite block tridiagonal generator matrices, values of nonzero entries of submodel transition matrices should be available during computation. In many cases, submodel transition matrices have subdiagonal, diagonal, or superdiagonal nonzero structures. If two submodel transition matrices associated with the same transition class have the same nonzero structure, then it is possible for the two matrices to share the storage space of one vector. In this case, a vector of size equal to the larger state space size of the two submodels is to be allocated. Besides, when $X_{0,1}^{(h)} = 1$ and $X_{p,l}^{(h)} = 1$ for $h = 1, \ldots, H$, $l = 1, \ldots, H_I$, and $p > Low$ as in some models, memory required to store nonzero entries of submodel transition submatrices at a given level is smaller than that of a higher level. In this case, it is feasible to allocate memory to store submatrices once at the highest level and keep reusing it when moving from level $High$ down to Low. Otherwise, memory necessary to store nonzero entries may be allocated and deallocated on the fly. Furthermore, vector–Kronecker product multiplication requires an additional vector over vector–matrix multiplication. When $\tilde{R}_l$ and A_l are stored as sparse matrices, an additional temporary vector is used to compute, compact, and store the rows of A_l. Besides, adding a row of a matrix in Kronecker form to a vector requires two additional vectors. We also choose to store the values of transition rate functions for states in levels $l-1$,l,$l+1$ when processing level l in order not to evaluate the functions multiple times. We allocate all the additional vectors at the outset and deallocate them at the end. Amount of memory allocated for all these vectors is negligible compared to the total amount of memory allocated for $\tilde{R}_l$ matrices.

If memory is at a premium, one can also consider the more recent approach in [21] that reduces memory requirements further by enabling the computation of steady-state expectations without having to obtain the steady-state distribution. The approach is inspired by a Horner-like computational scheme in which only the conditional expected sojourn time matrix $\tilde{R}_l$ at level l needs to be allocated in step l; otherwise, there are no time savings obtained. In other words, not all $\tilde{R}_l$ matrices for $l = High - 1$ down to Low need to be stored simultaneously. In order to evaluate M different functions of the steady-state distribution, $(M+1)$ temporary vectors of size equal to the num-

ber of states within levels *Low* through *High* must be used. For instance, if the first moment (i.e., mean) is to be computed for $M = H_I$ countably infinite variables, $(H_I + 1)$ temporary vectors of length $|\tilde{\mathcal{R}}|$ must be allocated. The additional vector is used for normalization purposes. At step l, $\tilde{R}_l$ is computed as usual and stored. This implies that $\tilde{R}_{l+1}$ from the previous step need not be in memory any longer, and hence, $\tilde{R}_l$ is the only conditional expected sojourn time matrix in memory at step l in this approach. Then $\tilde{R}_l$ is multiplied with the subvectors corresponding to level $(l+1)$ of the $(M+1)$ temporary vectors. The product is added to the running sum of subvectors corresponding to level l of the $(M + 1)$ temporary vectors in order to keep on accumulating steady-state expectations. With this memory efficient approach, steady-state measures based on average costs or rewards, moments, and cumulants can be computed. As shown in [18, 19], memory savings can be substantial with the approach in [21] in some models. This alternative solver is able to compute mean values of variables stably as the solver in Algorithm 9. The reported relative errors of the mean values obtained with the memory efficient solver with respect to those obtained with the original solver are close to machine precision in all cases. However, there is one drawback; it is the absence of the accuracy measure, $\|\boldsymbol{\pi}Q\|_1$, because the steady-state distribution, $\boldsymbol{\pi}$, is no longer available.

In between full and sparse storages of $\tilde{R}_l$ matrices, full storage is better in models having very high nonzero densities in the $\tilde{R}_l$ matrices. When there are memory savings with sparse storage of $\tilde{R}_l$ matrices [18, 19], the respective time savings are even more substantial. The temporary matrix A_l seems to be benefiting considerably from sparse LU factorization. On the other hand, there is observable difference between using sparse versus Kronecker representations of the $Q(l+1, l)$ matrices. This is the case because each subdiagonal nonzero block is used once, and the sparse generation procedure associated with it and the pre-multiplication with $\tilde{R}_{l+1}$ amount to performing the same number of flops as would be done by the vector–Kronecker product multiplication algorithm between the rows of $\tilde{R}_{l+1}$ and the subdiagonal nonzero block when the latter is kept in Kronecker form.

The scalability of different LDQBD solver implementations for increasing values of ε has been investigated in [18]. Since time complexity of A_l's LU factorization at level l is cubic in the order of $|\mathcal{R}_l|$ for dense $\tilde{R}_l$ matrices and $|\mathcal{R}_l|$ is a polynomial with degree $(l-1)$, time requirements become more pronounced for models with higher H_I values. The situation regarding memory is better since the requirements at level l are quadratic in $|\mathcal{R}_l|$ for dense $\tilde{R}_l$ matrices. Clearly, time and memory requirements are much better when the $\tilde{R}_l$ matrices are sparser. Given more memory and time, it is always possible to obtain more accurate results with the matrix analytic approach. Dropping nonzeros less than 10^{-16} from matrices is observed not to have an adverse effect on accuracy but likely to help with memory requirements [120].

In [103], an alternative technique for systems of stochastic chemical kinetics which is also based on using Lyapunov functions is investigated. The

technique explores states that are only in the set concentrating the steady-state probability mass and then resorts to polyhedra theory [86] to bound steady-state probabilities. Although the technique can potentially work on states arbitrarily far away from the origin, it yields results that are much less accurate compared to those provided by the technique discussed here.

In closing this section, we remark that the MAP/PH/S queueing system with acyclic PH retrials [112] considered in Chapter 4 could also be modeled as an LDQBD by choosing an appropriate level definition. However, the truncated model is represented as a block matrix with blocks of equal size using sums of Kronecker products, and its steady-state solution is computed iteratively with SOR. The reason for this choice is that the matrix analytic method requires the computation of the matrices of conditional expected sojourn times, and this computation does not scale well as the number of dimensions in the multidimensional MC increases. This is due to the increase in the order of the diagonal blocks as the LDQBD level number increases in multidimensional MCs. An implementation within the `NSolve` package of the APNN Toolbox [7, 22] is available at [114].

6.6 Working with Compact Solution Vectors

The HTD format for compact vectors discussed in the previous chapter is recently employed in power, JOR, and GMRES iterative methods using two adaptive truncation strategies for the solution vectors of MCs [66]. The performance of the resulting iterative solvers are compared on a large number of multidimensional problems, two of which are the availability and polling models in Chapter 2, with their Kronecker structured counterparts that employ full solution vectors of length $|\mathcal{R}|$, the size of the reachable state space. In this section, we briefly review the outcome of the study in [66].

Let us recall that the power method is successfully employed in the *PageRank* algorithm [41] for Google matrices [106], and JOR is essentially a preconditioned power method in which the preconditioner is $M := Q_D/\omega$ and, hence, diagonal. The convergence rate of these methods is known to depend on the magnitude of the subdominant eigenvalue of the corresponding iteration matrix. In the original PageRank algorithm, this value is set to 0.85 by construction to guarantee a certain convergence rate. Furthermore, JOR is known to converge for $\omega \in (0, 1)$, that is, under-relaxation, as discussed in Section 6.1. On the other hand, GMRES is a projection method which extracts solutions from an increasingly higher-dimensional Krylov subspace by enforcing a minimality condition on the residual 2-norm at the expense of having to compute and store a new *orthogonal basis vector* for the subspace at each iteration. This orthogonalization is accomplished through what is called an *Arnoldi process*. In theory, GMRES converges to the solution in at most $|\mathcal{R}|$ iterations. However, this may become prohibitively expensive at least in

terms of space, so in practice, a restarted version with a finite subspace size of m is to be used. Hence, the number of vectors allocated for the representation of the Krylov subspace in the *restarted* GMRES solver will be limited by m. If the vectors are stored in compact format, m can be set to a larger value without exceeding the available memory which is expected to improve the convergence of GMRES.

Unlike the power and GMRES methods, the iteration vector also needs to be multiplied by the reciprocated diagonal elements of the coefficient matrix in the JOR method. Since it is costly to generate each element of a vector in HTD format, an iterative method such as *Newton–Schulz* [202] can be used to perform the elementwise reciprocation of the diagonal vector in HTD format at the outset using Algorithm 10.

ALGORITHM 10. *Elementwise reciprocation of vector* $\mathbf{d}$ *as* $\mathbf{dinv}$.

$\mathbf{dinv} := \mathbf{e}^T/\|\mathbf{d}\|_2$; $it := 0$; $nrm := 0$; $nrm\_chg := 0$; $rcp\_stop := FALSE$;
While $nrm \geqslant$ `rcp_tol` and $rcp\_stop = FALSE$,
    $it := it + 1$;
    $\mathbf{dinv}' := \mathbf{dinv}$;
    $\mathbf{\Delta d} := \mathbf{e}^T - \mathbf{d} \star \mathbf{dinv}'$;
    $nrm' := \|\mathbf{\Delta d}\|_2/\|\mathbf{e}\|_2$;
    $\mathbf{dinv} := \mathbf{dinv}' + \mathbf{dinv}' \star \mathbf{\Delta d}$;
    If $it > 1$,
        $nrm\_chg := nrm'/nrm$;
    If $nrm\_chg >$ `rcp_chg_tol` and $it \geqslant$ `rcp_maxit`,
        $rcp\_stop := TRUE$;
    Else
        $nrm := nrm'$.

The algorithm starts by initializing $\mathbf{dinv}$ (which becomes $\mathbf{dinv}'$ in the current iteration) with a vector whose entries are equal to each other in the same vein as the initialization of the starting vector in iterative methods with the uniform distribution. This step requires the computation of the 2-norm of $\mathbf{d}$. The other operations that the algorithm executes are elementwise multiplication of two vectors which is denoted by the operator $\star$, addition of two vectors, and computation of the 2-norm of vectors. The algorithm stops if the 2-norm of $(\mathbf{e}^T - \mathbf{d} \star \mathbf{dinv}')$ relative to the 2-norm of $\mathbf{e}$ is smaller than `rcp_tol` [202] or if this relative norm divided by the relative norm in the previous iteration is greater than or equal to `rcp_chg_tol` when `rcp_maxit` iterations have already been performed. We remark that the Newton–Schulz method is a nonlinear iteration with quadratic convergence rate [185].

Among the operations needed to implement Algorithm 10, elementwise multiplication of two compact vectors in HTD format has not been discussed in Section 5.3. Note that this operation is required not only in the Newton–Schulz iteration for computing the reciprocated diagonal elements in $\mathbf{dinv}$ at

the outset but also in multiplying **dinv** with the updated solution vector at each JOR iteration. Let us now explain how this operation can be performed.

Elementwise multiplication of two matrices Y and X with SVDs $Y = U_Y \Sigma_Y V_Y^T$ and $X = U_X \Sigma_X V_X^T$ yields

$$Y \star X := (U_Y \odot^T U_X)(\Sigma_Y \otimes \Sigma_X)(V_Y \odot^T V_X)^T,$$

where $\odot^T$ denotes a transposed variant of the *Khatri–Rao product* [199]. More specifically, the ith row of the $(n_{AB} \times r_A r_B)$ matrix $A \odot^T B$ is the Kronecker product of the ith rows of the $(n_{AB} \times r_A)$ matrix A and the $(n_{AB} \times r_B)$ matrix B. This implies that elementwise multiplication $Y \star X$ has rank equal to the product of the ranks of Y and X. Elementwise multiplication of two vectors $\mathbf{y}$ and $\mathbf{x}$ in HTD format is no different if SVD is replaced with HOSVD. The operation can be carried out in four steps [202], which are orthogonalization of the vectors in HTD format, computation of their Gramians, computation of SVDs of Gramians, and update of basis and transfer matrices. There is also a truncation step that needs to be performed when the elementwise multiplication is extracted from the implicitly formed Kronecker product by again imposing an accuracy of `trunc_tol` on the truncated HOSVD.

The elementwise reciprocation of $\mathbf{d}$ using Newton–Schulz introduces two different truncations [202], first, before Newton–Schulz, when the HTD of the diagonal elements in $\mathbf{d}$ is computed, and, second, during the iterative computation of the reciprocated diagonal elements in **dinv** within Newton–Schulz. Since a numerical iterative method is employed to compute the reciprocal values, it is not clear how fast convergence takes place in practice since the initial transient period that is needed for the asymptotic quadratic convergence behavior to manifest itself can be time consuming [185]. Therefore, an alternative approach is also considered in [66].

The alternative approach is to take advantage of identical entries in $\mathbf{d}(\mathcal{R}_p)$ for $p = 0, \ldots, N-1$. In many multidimensional Markovian models, there are entries with the same value in $\mathbf{d}(\mathcal{R}_p)$ because the CTMC results from some compact model specification which contains only a few parameters compared to the size of the reachable state space, $\mathcal{R}$. This may be exploited by defining an equivalence relation among the states in $\mathcal{R}_p^{(h)}$ for $h = 1, \ldots, H$ by using row sums of the submatrices $Q_k^{(h)}(\mathcal{R}_p, \mathcal{R}_w)$ that contribute to sums of Kronecker products for $k \in \mathcal{K}_{p,w}$ and $w = 0, \ldots, N-1$. Then the states in $\mathcal{R}_p$ that are in the same equivalence class in all dimensions have identical diagonal entries in $\mathbf{d}(\mathcal{R}_p)$ [66]. This approach can be very effective in representing $\mathbf{d}(\mathcal{R}_p)$ compactly when the number of equivalence classes in each dimension is small. All that needs to be done is to represent the reciprocated diagonal entries in each equivalence class using a Kronecker product, which in turn can be represented in HTD format. Thus, for each combination of equivalence classes across all dimensions, an HTD formatted vector must be constructed. Then all such vectors need to be added and truncated so that the reciprocated diagonal elements in $\mathbf{dinv}(\mathcal{R}_p)$ are represented by a single HTD formatted

vector. In the worst case, one vector in HTD format needs to be constructed for each state meaning all entries in $\mathbf{d}(\mathcal{R}_p)$ are different from each other. However, for many models, the number of HTD formatted vectors to be added and truncated is much smaller.

In `htucker`, memory requirements of the HTD format are limited by a fixed truncation error tolerance, `trunc_tol`, and a fixed rank bound, `max_rank`, for the truncated HOSVD. In many cases, this approach results in ranks becoming large during the first few iterations when the iteration vector is not close to the solution. Furthermore, imposing a fixed rank bound may also limit the accuracy of the final solution that can be obtained. Therefore, it is more efficient to use a larger `trunc_tol` when the 2-norm of the residual vector, $\|\mathbf{r}_{(it)}\|_2$, is large and to decrease `trunc_tol` when $\|\mathbf{r}_{(it)}\|_2$ is becoming small. Thus, an *adaptive strategy* for adjusting the truncation error is required, and two such strategies are considered in [66].

In the first adaptive strategy named AS1, `trunc_tol`, is initially set to $\texttt{trunc\_tol} := \texttt{stop\_tol}/\sqrt{2H-3}$ and then updated when $\|\mathbf{r}_{(it)}\|_2$ is computed at iteration *it* as

$$\begin{aligned}&\texttt{prev\_trunc\_tol} := \texttt{trunc\_tol},\\ &\quad\texttt{trunc\_tol} := \max(\min(\texttt{prev\_trunc\_tol}, \sqrt{\|\mathbf{r}_{(it)}\|_2}\ \texttt{stop\_tol}/\sqrt{2H-3}),\\ &\qquad\qquad 10^{-16}).\end{aligned}$$

Initially, $\texttt{prev\_trunc\_tol} := 0$. In this way, `trunc_tol` is made to remain within $(10^{-16}, \texttt{prev\_trunc\_tol})$ while being forced to decrease conservatively with a decrease in $\|\mathbf{r}_{(it)}\|_2$.

In the second adaptive truncation strategy named AS2, we start with the initialization $\texttt{prev\_trunc\_tol} := 0$ and $\texttt{trunc\_tol} := 100\ \texttt{stop\_tol}/\sqrt{2H-3}$. Then we update the two variables as

$$\begin{aligned}&\texttt{prev\_trunc\_tol} := \texttt{trunc\_tol},\\ &\quad\texttt{trunc\_tol} := \max(\min(\texttt{prev\_trunc\_tol}, \sqrt{\|\mathbf{r}_{(it)}\|_2}\ 10^{-4}/\sqrt{2H-3}),\\ &\qquad\qquad 10^{-16}).\end{aligned}$$

when $\|\mathbf{r}_{(it)}\|_2$ is computed at iteration *it*. AS2 starts at a larger `trunc_tol` and is expected to increase `trunc_tol` more conservatively compared to AS1 and independently of the value of `stop_tol`.

The scalability of the HTD format in the compact vector iterative solvers of power, JOR, and GMRES using the above adaptive truncation strategies is investigated on three models for increasing sizes of the reachable state space. Two of the models are the availability and polling models in Chapter 2, and the third one is a cloud computing model from [154]. First, we summarize the effect of using the Newton–Schulz iteration and the equivalence class approach to compute **dinv** at the outset for JOR.

Each of the availability and cloud computing models has a single reachable state space partition and ends up enumerating a large percentage of the entries in **d** to construct **dinv**, since the number of equivalence classes in dimension h of both models is relatively large with respect to the state space size $|\mathcal{S}^{(h)}|$ for $h = 1, \ldots, H$. It is observed that the time to compute **d** and therefore **dinv** in HTD format grows quickly for larger values of H in these two models using both approaches. However, the Newton–Schulz approach can be made faster by limiting `rcp_maxit` in Algorithm 10. At the moment, it starts with truncated HOSVD ranks of 1 and increases the ranks when needed as the iterations progress. For these two models, the equivalence class approach for computing **dinv** in HTD format yields a more compact representation than the Newton–Schulz approach. As for the polling model which has multiple reachable state space partitions, the number of equivalence classes used to represent **d** is small. Hence, this approach yields not only the more compact representation but also the faster one. In this case, the HTD representation of **dinv** remains compact even for larger values of H, with its memory requirements growing more or less linearly in H.

Solution vectors of the Kronecker-based solvers power and JOR using the HTD format cease to be unit 1-norm due to truncation with `trunc_tol` and are normalized at each iteration. The check on `stop_tol` is also performed at each iteration in this case. For Kronecker-based power and JOR solvers with full vectors, normalization and the stopping test can take place every some number of (e.g., 10) iterations. The parameters of solvers used in the experiments are $\Delta := 0.999/\max_{i \in \mathcal{R}} |q_{i,i}|$ for power, $\omega := 0.75$ for JOR, $m := 30$ for the Krylov subspace size of GMRES, `stop_tol` $:= 10^{-8}$, and `max_time` := 1,000 seconds. Detailed results can be found in [66].

Memory requirements of Kronecker-based full-vector solvers can be calculated at the outset. Power and JOR each require three vectors of length $|\mathcal{R}|$ (for Q_D, $\boldsymbol{\pi}_{(it)}$, and $\boldsymbol{\pi}_{(it+1)}$) and two vectors of length $\max_p |\mathcal{R}_p|$ (for the shuffle algorithm to carry out the full vector–Kronecker product multiplication), whereas restarted GMRES(m) requires $(m + 3)$ vectors of length $|\mathcal{R}|$ and two vectors of length $\max_p |\mathcal{R}_p|$. Clearly, largest versions of the three models cannot be handled with full-vector solvers on a platform having 16 GB of main memory. Similarly, the Kronecker-based full-vector GMRES solver cannot be utilized in the six-dimensional polling model. For Kronecker-based compact vector solvers, it is not possible to forecast memory requirements at the outset when adaptive truncation strategies are used. This is because the maximum number of floating-point array elements allocated to matrices in the HTD format representing vectors and the workspace used in the solution process depends on the values in the vectors that are represented compactly at each iteration, and, hence, the character of the particular model and the behavior of the solver, and also on the value of `trunc_tol`.

There are cases where we have not been able to obtain **dinv** in HTD format for JOR(0.75) within `max_time` or due to memory limitations sometimes with both computational approaches and sometimes with Newton–Schulz alone.

On the other hand, there is a case where the Kronecker-based compact vector power method with AS2 exits due to a non-converging `LAPACK` method that is used to compute SVD. Similarly, there are a few cases where we have not been able to obtain results with the Kronecker-based compact vector GMRES(30) solver using AS1. The solver performs a number of Arnoldi process steps and then takes an exceedingly large amount of time without producing a result and therefore is aborted. Finally, there is a case where the Kronecker-based compact vector JOR(0.75) solver with AS1 using the equivalence class approach for reciprocation fails due to memory limitations.

Among the Kronecker-based full-vector solvers for all three models, JOR is almost always the fastest solver yielding the smallest $\|\mathbf{r}_{(it)}\|_2$ with the smallest memory, which is about one order of magnitude smaller than what is required by GMRES. Among the compact vector solvers, those that use AS1 never require a larger number of iterations than their counterparts with AS2 for the same $\|\mathbf{r}_{(it)}\|_2$. The average time per iteration of compact vector solvers with AS2 is lower than that with AS1. However, power method with AS2 stagnates and is not able to meet `stop_tol` in any of the problems. The convergence behavior of compact vector solvers employing AS2 tend to be more unpredictable and less satisfactory than that with AS1. Whenever a compact vector solver using AS1 stops, $\|\mathbf{r}_{(it)}\|_2 <$ `stop_tol`, whereas this inequality in general does not hold for AS2. Regarding memory requirements however, AS2 is better, sometimes by several orders of magnitude, than the respective solver that uses AS1. One would expect this to imply that as the number of dimensions increases, a larger number of iterations can be performed by AS2 in the same duration, thereby bringing the solver closer, if not, to convergence. This seems to be the case especially when the decrease in memory consumption is substantial. The full-vector approach is faster for smaller models, but it is outperformed by the compact vector approach for larger models especially when it is used with the power method and AS2.

The availability model has highly unbalanced transition rates due to infrequent failures. Failures correspond to local transitions, and since they are infrequent, the system is available most of the time, and the steady-state distribution is skewed. Effects of varying the original transition rates given in Section 5.3 for $H = 6$ are investigated on two variants. The first variant has one tenth, and the second variant has ten times the failure rates of the original model. The steady-state distribution becomes more skewed in the first variant since the system is even more available in the long run. This translates into a less difficult problem to solve, meaning it takes a smaller number of iterations for the same accuracy of the solution. The results show that indeed all solvers are sensitive to the transition rates in the model. Memory requirements and computation time of **dinv** increase in the more difficult variant. Memory requirements of power and JOR(0.75) compact solvers increase when the problem becomes more difficult, resulting in longer time per iteration and a smaller number of iterations in the same time duration.

In summary, the compact vector approach when coupled with an adaptive truncation strategy is memory and relatively time efficient, having the potential to increase the size of solvable models on a given platform significantly, and therefore deserves further investigation.

Chapter 7
Transient Analysis

Transient analysis for MCs refers to computing the probability distribution at a particular time instant starting from an initial probability distribution. As such, steady state need not exist to carry out transient analysis, and the more interesting problem formulation turns out to be related to CTMCs.

For a CTMC with reachable state space $\mathcal{R}$ and generator matrix Q in Kronecker form, let $\boldsymbol{\pi}_0 \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}$ be the probability distribution at time 0 with $\boldsymbol{\pi}_0\mathbf{e} = 1$. Then the transient probability distribution vector $\boldsymbol{\pi}_t \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}$ of the CTMC at time $t \in \mathbb{R}_{\geqslant 0}$ with $\boldsymbol{\pi}_t\mathbf{e} = 1$ is given by

$$\boldsymbol{\pi}_t := \boldsymbol{\pi}_0 e^{Qt}. \tag{7.1}$$

Transient solution methods are based on computing the *matrix exponential* e^{Qt} explicitly or implicitly [305]. Since computing e^{Qt} explicitly is not feasible when Q is large, sparse, and in Kronecker form, methods that are able to follow the latter approach are considered. These are uniformization [164, 169, 275], projection methods [261, 294], and ordinary differential equation (ODE) solvers [164, 275, 294]. Among these solvers, projection methods require the computation of a polynomial approximation associated with the Krylov subspace which is tackled using dense methods. Furthermore, projection methods require the use of a large number of supplementary vectors as previously discussed for steady-state analysis. These are shortcomings of projection methods as transient solvers not only for generator matrices in Kronecker form but also for compact solution vectors. Therefore, in this chapter we concentrate on uniformization and ODE solvers.

The equivalent problem for a DTMC with reachable state space $\mathcal{R}$ and transition probability matrix P in Kronecker form is to compute the probability distribution at a particular step starting from an initial probability distribution $\boldsymbol{\pi}_{(0)} \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}$ at step 0 with $\boldsymbol{\pi}_{(0)}\mathbf{e} = 1$. In this case, the transient

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_7

probability distribution vector $\boldsymbol{\pi}_{(m)} \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}$ of the DTMC at step $m \in \mathbb{Z}_{\geqslant 0}$ with $\boldsymbol{\pi}_{(m)}\mathbf{e} = 1$ is given by

$$\boldsymbol{\pi}_{(m)} := \boldsymbol{\pi}_{(0)} P^m,$$

where P^m is the m–step transition probability matrix. The form of the solution suggests using vector–Kronecker product multiplications to compute the product $\boldsymbol{\pi}_{(it)}P$ in

$$\boldsymbol{\pi}_{(it+1)} := \boldsymbol{\pi}_{(it)}P \quad \text{for } it = 0, \ldots, m-1$$

when P is large, sparse, and in Kronecker form to obtain $\boldsymbol{\pi}_{(m)}$. In other words, the transient distribution of a DTMC is given by the power method with iteration matrix P. For finite irreducible DTMCs, this requires P to be aperiodic, hence ergodic, for convergence to steady state. Note that convergence to the steady-state vector $\boldsymbol{\pi}$ in computer precision may take place before reaching the mth step when P is ergodic. Therefore, it is customary to execute a test on the error or residual norm of the solution, say, every 10 steps so that the iteration can be terminated earlier if convergence is detected.

We start with uniformization, a very simple method for the computation of the transient distribution vector of CTMCs in Kronecker form.

7.1 Uniformization

Given a CTMC with generator matrix Q as in (2.10), *uniformization* [164, 169, 275] considers the uniformized transition probability matrix

$$P := I + \frac{1}{\Gamma}Q, \quad \Gamma \geqslant \max_{i\in\mathcal{R}} |q_{i,i}|$$

scaled by a rate Γ that is greater than or equal to the largest exit rate $\max_{i\in\mathcal{R}} |q_{i,i}|$ among the states in $\mathcal{R}$, that is,

$$\Gamma P = \Gamma I + Q,$$

so that transitions occur at the same (uniform) rate of Γ in each state. Here, $\max_{i\in\mathcal{R}} |q_{i,i}|$ is the smallest value of Γ for which P is a stochastic matrix.

The uniformized DTMC underlying P can be viewed as a discretized continuous-time process associated with the CTMC that is embedded in a Poisson process of rate Γ, since the time to exit each state is exponentially distributed with rate Γ. Then the uniformization equation at time $t \in \mathbb{R}_{\geqslant 0}$ is given by

$$e^{Qt} = \sum_{m=0}^{\infty} e^{-\Gamma t} \frac{(\Gamma t)^m}{m!} P^m$$

from the Maclaurin series (i.e., Taylor series around 0) expansion of the matrix exponential e^{Qt} after substituting $Q = \Gamma P - \Gamma I$ (see also [90]). Observe in e^{Qt} that

$$Prob(m \text{ transitions in time } t) := e^{-\Gamma t}\frac{(\Gamma t)^m}{m!}$$

is the Poisson probability of having m transitions during a time interval of length t in which each transition takes place at rate Γ and P^m is the m–step transition probability matrix, hence also the name *randomization*.

The infinite series corresponding to the matrix exponential can be truncated, say, at a maximum number of M_r transitions within time t and used to compute an approximation to $\boldsymbol{\pi}_t$ from (7.1) as in

$$\tilde{\boldsymbol{\pi}}_t := \boldsymbol{\pi}_0 \sum_{m=0}^{M_r} \mathrm{e}^{-\Gamma t}\frac{(\Gamma t)^m}{m!}P^m \ . \tag{7.2}$$

Equation (7.2) is the simplest way by which the transient solution can be stably computed since it involves only positive floating-point arithmetic and therefore avoids subtractive cancellation. The approach is originally proposed in [191] and also called *Jensen's method*.

The uniformization method has the known truncation error [164]

$$\sum_{m=0}^{M_r}\frac{(\Gamma t)^m}{m!} \geqslant \frac{1-\epsilon}{e^{-\Gamma t}} \quad \text{implying} \quad \|\boldsymbol{\pi}_t - \tilde{\boldsymbol{\pi}}_t\| \leqslant \epsilon$$

obtained from the fact that Poisson probabilities sum to 1. The number of terms given by the right truncation point M_r to satisfy the truncation error bound ϵ increases linearly with Γt. With the increase in Γt, the Poisson probabilities for smaller values of m start to become negligible. In such a situation, it is useful to have also a left truncation point, M_l, to save on computations. This modification in uniformization yields

$$\tilde{\boldsymbol{\pi}}_t := \sum_{m=M_l}^{M_r} \mathrm{e}^{-\Gamma t}\frac{(\Gamma t)^m}{m!}\boldsymbol{\pi}_0 P^m \ .$$

The Poisson probabilities can be computed stably by the algorithm in [148]. Therein, left truncation is suggested for $\epsilon := 10^{-10}$ when $\Gamma t \geqslant 25$ and the number of terms $(M_r - M_l + 1)$, required to satisfy a fixed truncation error is shown to be proportional to $\sqrt{\Gamma t}$. To minimize the rounding error that is incurred by adding small numbers to large ones when summing the probabilities for normalization, an approach that performs the summation from both ends toward the mode is used so that small numbers are summed with small numbers first.

Once the Poisson probabilities are available, a Horner-like implementation of uniformization would be

$$\mathbf{p} := \mathrm{e}^{-\Gamma t}\frac{(\Gamma t)^{M_l}}{M_l!}\boldsymbol{\pi}_0 P^{M_l}, \quad \tilde{\boldsymbol{\pi}}_t := \mathbf{p},$$

$$\mathbf{p}' := \mathbf{p}P, \quad \mathbf{p} := (\Gamma t/m)\mathbf{p}', \quad \tilde{\boldsymbol{\pi}}_t := \tilde{\boldsymbol{\pi}}_t + \mathbf{p} \quad \text{for } m := M_l + 1, \ldots, M_r.$$

The initialization of $\mathbf{p}$ requires M_l multiplications of $\boldsymbol{\pi}_0$ with P using efficient vector–Kronecker product multiplication followed by scaling of the resulting vector with $Prob(M_l$ transitions in time $t)$. Thereafter, for $m = M_l + 1, \ldots, M_r$ step m consists of the multiplication of $\mathbf{p}$ with P using efficient vector–Kronecker product multiplication to obtain $\mathbf{p}'$ followed by the scaling of $\mathbf{p}'$ with $\Gamma t/m$ to obtain $\mathbf{p}$ and the addition of $\mathbf{p}$ to the running sum $\tilde{\boldsymbol{\pi}}_t$. This requires three vectors of length $|\mathcal{R}|$ for $\boldsymbol{\pi}_t$, $\mathbf{p}$, $\mathbf{p}'$ and two vectors of length $\max_p |\mathcal{R}_p|$ to carry out the efficient vector–Kronecker product multiplication. The uniformization method is implemented within the `NSolve` package of the APNN toolbox [7, 22].

The disadvantage of uniformization is the need to partition t into time steps as in multistep ODE methods and use a smaller ϵ for each subinterval when Γt is large [305]. In passing to ODE solvers, we remark that existence of state exit rates, $|q_{i,i}|$ for $i \in \mathcal{R}$, belonging to different time scales in a CTMC coupled with the choice of t proportional to the reciprocal of the smallest exit rate, that is, $t \propto 1/\min_{i\in\mathcal{R}} |q_{i,i}|$, implies *stiffness* in the corresponding problem [275]. Note that this definition of stiffness can be interpreted as the largeness of the ratio $\max_{i\in\mathcal{R}} |q_{i,i}|/\min_{i\in\mathcal{R}} |q_{i,i}|$ when $t \propto 1/\min_{i\in\mathcal{R}} |q_{i,i}|$, and the performance of uniformization degrades when the problem becomes stiff.

7.2 Ordinary Differential Equation Solvers

Numerical ODE methods to solve (1.2) attempt to follow a unique solution curve from an initial value of the transient probability distribution vector $\boldsymbol{\pi}_t \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}$ at time $t = 0$ with $\boldsymbol{\pi}_t\mathbf{e} = 1$ to its value at some other prescribed time $t \in \mathbb{R}_{\geqslant 0}$ [305]. Thus, an *initial value problem* [12] involving a system of linear first-order ODEs as in

$$\frac{d\boldsymbol{\pi}_t}{dt} = \mathbf{f}(t, \boldsymbol{\pi}_t) \quad \text{with} \quad \mathbf{f}(t, \boldsymbol{\pi}_t) = \boldsymbol{\pi}_t Q \quad \text{given} \quad \boldsymbol{\pi}_0 \in \mathbb{R}_{\geqslant 0}^{1\times|\mathcal{R}|}, \ \boldsymbol{\pi}_0\mathbf{e} = 1 \quad (7.3)$$

is to be solved.

The solution employs discretization on the interval $[0, t]$ and approximations to the solution at intermediate time steps. Given a discrete set of time steps t_m for $m = 0, \ldots, M$ with $t_0 = 0$ and $t_M = t$ in $[0, t]$ and $\Delta t_m = t_m - t_{m-1}$ as the *step size* of step m, the exact solution of (7.3) at time t_m is denoted by $\boldsymbol{\pi}_{t_m}$. A numerical ODE method computes an approximation $\boldsymbol{\pi}_{(m)}$ to $\boldsymbol{\pi}_{t_m}$ for $m = 1, \ldots, M$ starting from the initial condition $\boldsymbol{\pi}_{(0)} = \boldsymbol{\pi}_0$.

In computing the approximation $\boldsymbol{\pi}_{(m+1)}$, an ODE method may incorporate the values of previously computed approximations $\boldsymbol{\pi}_{(l)}$ for $l = 0, \ldots, m$ or even use an approximation of $\boldsymbol{\pi}_{t_{m+1}}$. A method that uses only $(t_m, \boldsymbol{\pi}_{(m)})$ to compute $\boldsymbol{\pi}_{(m+1)}$ is said to be an *explicit single-step* method. It is said to be a *multistep* method if it uses approximations at several previous steps to compute its new approximation. A method is said to be *implicit* if computation of $\boldsymbol{\pi}_{(m+1)}$ requires an approximation to $\boldsymbol{\pi}_{t_{m+1}}$; otherwise, it is said to be *explicit* [305].

In the next two sections, we present examples of single-step and multistep methods. Then an explicit single-step method will be used to initialize the first few approximations of an implicit multistep method to compute the transient solution of a countably infinite system of stochastic chemical kinetics known to be stiff.

7.2.1 Runge–Kutta Methods

Runge–Kutta (RK) methods [12, 305] are single-step methods that may be explicit or implicit. An RK method of *order* o, denoted RKo, provides an accuracy comparable to a Taylor series approximation to $\boldsymbol{\pi}_{(m+1)}$ of order o around $\boldsymbol{\pi}_{(m)}$, however without having to determine and evaluate the derivatives $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \ldots, \mathbf{f}^{(o-1)}$. Instead, for a step size of Δt, the evaluation of

$$\begin{aligned}\mathbf{k}_{(s)} &= \mathbf{f}\left(t_m + c_s \Delta t \boldsymbol{\pi}_{(m)} + \Delta t \sum_{s'=1}^{S} a_{s,s'} \mathbf{k}_{(s')}\right) \\ &= \left(\boldsymbol{\pi}_{(m)} + \Delta t \sum_{s'=1}^{S} a_{s,s'} \mathbf{k}_{(s')}\right) Q \quad \text{for } s = 1, \ldots, S\end{aligned}$$

at S selected points $t_m + c_s \Delta t$ called *stages* between t_m and $t_{m+1} (= t_m + \Delta t)$ is performed, so that the approximation at time step t_{m+1} is computed from

$$\boldsymbol{\pi}_{(m+1)} := \boldsymbol{\pi}_{(m)} + \Delta t \sum_{s=1}^{S} b_s \mathbf{k}_{(s)}.$$

The derivation of $\mathbf{b} \in \mathbb{R}^{S \times 1}$ and $A \in \mathbb{R}^{S \times S}$ with $\mathbf{c} = A\mathbf{e}$ whose entries are used at each stage of RKo is obtained from a comparison with the terms through Δt^o in the Taylor series approximation [12] for the first step, that is, the computation of $\boldsymbol{\pi}_{(1)}$ from the initial condition $(t_0, \boldsymbol{\pi}_{(0)})$ [305]. The order conditions that must be satisfied by RK methods are given in [12]. These methods are explicit for strictly lower-triangular A and implicit otherwise. In the former case, $\mathbf{k}_{(s)}$ can be computed by vector–Kronecker product multiplications when Q is in Kronecker form and therefore the method is relatively easy to use. In the latter case, there will be dependencies between the left- and right-hand sides of the equation for $\mathbf{k}_{(s)}$ implying a more costly implementation to obtain the solution at each step.

An important issue pertains to the choice of step size. The step size should be sufficiently small so that the desired accuracy of the computed solution is attained but large enough to avoid redundant computations when obtaining the solution [305]. The error that is made in single-step methods such as RK within one step assuming that the solution obtained at the previous step was exact is named as being *local*. For RKo, local error due to truncation of the Taylor series [305] is given by

$$\|\boldsymbol{\pi}_{t_0+\Delta t} - \boldsymbol{\pi}_{(1)}\| = C\Delta t^{o+1} + O(\Delta t^{o+2}) \quad \text{for some } C \in \mathbb{R}_{\geqslant 0}.$$

Note that this result does not include any rounding errors that may accrue during the computation. On the other hand, *global error* can be defined as the error in the computed solution at time t, that is, $\|\boldsymbol{\pi}_t - \boldsymbol{\pi}_{(M)}\|$.

Keeping the step size constant throughout the solution process is not a good idea since the computed solution $\boldsymbol{\pi}_{(m)}$ may exhibit different variations in different parts of the solution interval $[0, t]$. It is preferable to change the step size so that the local error is roughly equal to a user-specified tolerance, `stop_tol`, at each step. Many times this tolerance is multiplied by a safety factor less than but close to 1 so that the step size change is carried out more conservatively [12]. In order to obtain an estimate of the local error, it is customary to compute two approximations $\boldsymbol{\pi}_{(m)}$ and $\hat{\boldsymbol{\pi}}_{(m)}$ to the solution at step m and take $\|\hat{\boldsymbol{\pi}}_{(m)} - \boldsymbol{\pi}_{(m)}\|$ as an estimate of the local error. If

$$\|\hat{\boldsymbol{\pi}}_{(m)} - \boldsymbol{\pi}_{(m)}\| < \texttt{stop\_tol}$$

does not hold, then Δt is discarded. A new step size Δt^* is computed from

$$\Delta t^* := \Delta t \left(\frac{\texttt{stop\_tol}}{\|\hat{\boldsymbol{\pi}}_{(m)} - \boldsymbol{\pi}_{(m)}\|} \right)^{\frac{1}{o+1}},$$

and the computation for step m is repeated until an acceptable step size is found [12, 305]. Runge–Kutta–Fehlberg (RKF) we discuss next is one such method that can be used to this end.

At step m, the *embedded* RK method due to Fehlberg [133] uses two approximations of different orders, $\boldsymbol{\pi}_{(m)}$ for estimating the solution with order o and $\hat{\boldsymbol{\pi}}_{(m)}$ for estimating the local error in the solution with order $o + 1$ [12, 305]. The RKo method uses A, $\mathbf{b}$, and $\mathbf{c}$, while the RK$(o + 1)$ method uses A, $\hat{\mathbf{b}}$, and $\mathbf{c}$. Because RKo is embedded inside RK$(o + 1)$, the two approximations can share stage computations.

The RKF method, written RFK$o(o+1)$, with $o = 4$ is used in [275] for the transient analysis of CTMCs. For RKF4(5), we have the six-stage single-step method of order 4 with (or a method of order 5 without) an error estimate [12] using

$$A = \begin{pmatrix} & & & & & \\ \frac{1}{4} & & & & & \\ \frac{3}{32} & \frac{9}{32} & & & & \\ \frac{1,932}{2,197} & -\frac{7,200}{2,197} & \frac{7,296}{2,197} & & & \\ \frac{439}{216} & -8 & \frac{3,680}{513} & -\frac{845}{4,104} & & \\ -\frac{8}{27} & 2 & -\frac{3,544}{2,565} & \frac{1,859}{4,104} & -\frac{11}{40} & \end{pmatrix}, \quad b = \begin{pmatrix} \frac{25}{216} \\ \\ \frac{1,408}{2,565} \\ \frac{2,197}{4,104} \\ -\frac{1}{5} \\ \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} \frac{16}{135} \\ \\ \frac{6,656}{12,825} \\ \frac{28,561}{56,430} \\ -\frac{9}{50} \\ \frac{2}{55} \end{pmatrix}.$$

Observe that A is strictly lower-triangular.

It is difficult to estimate the global error [12, 305]. Often multiple solutions with different values of `stop_tol` are obtained, and the solutions are compared to provide an estimate of the global error. On the other hand, one can always use the sum of local errors as an upper bound on the global error; however, the resulting bound may not be satisfactory. For stiff systems the step size, Δt, may need to be intolerably small before acceptable accuracy is obtained, resulting in unacceptable rounding errors and computation time requirements. In general, explicit methods do not work for stiff problems.

7.2.2 Backward Differentiation Formulae

Backward differentiation formulae (BDF) are a class of multistep methods that are widely used to cope with stiffness in the solution of ODEs [152]. In this class of solvers, time is discretized, and solutions at intermediate time steps are computed using *backward differences*. We remark that the lth backward difference [12] approximates the lth derivative $\mathbf{f}^{(l)}$ times Δt^l for step size $\Delta t > 0$. Only implicit BDF methods are popular due to the fact that explicit versions can be used solely with one or two steps and are not effective [12, 305]. In this section, we will be referring to implicit BDF methods as BDF methods without using the qualifier implicit.

BDF methods are constructed by differentiating a degree o *interpolating polynomial* $\mathbf{z}_t$ that passes through the $o+1$ points $(t_l, \boldsymbol{\pi}_{(l)})$ for $l = m+1-o, \ldots, m+1$. Furthermore, $\boldsymbol{\pi}_{(m+1)}$ is determined so that $\mathbf{z}_t$ satisfies the ODE at time step t_{m+1}, that is, $d\mathbf{z}_{t_{m+1}}/dt_{m+1} = \boldsymbol{\pi}_{(m+1)}Q$ [305]. Hence, the BDF method is o-step because it uses the previous o approximations $\boldsymbol{\pi}_{(m+1-o)}, \ldots, \boldsymbol{\pi}_{(m)}$ at step $m+1$, it is implicit because it uses an approximation to $\boldsymbol{\pi}_{t_{m+1}}$ when computing $\boldsymbol{\pi}_{(m+1)}$, and it is of order o because the linearity of the method in $\mathbf{f}$ implies a local truncation error that is proportional to Δt^o [12].

Now, let $\boldsymbol{\pi}_{(m+1-l)}$ be the approximation to the transient probability distribution vector of the model at time step $t_{m+1-l} (= t_{m+1} - l\Delta t)$ for $l = 0, \ldots, o$. Then, the BDF method of order o, denoted BDFo, at step $m+1$ is given by

$$\sum_{l=1}^{o} \frac{1}{l} \nabla^l \boldsymbol{\pi}_{(m+1)} = \Delta t \boldsymbol{\pi}_{(m+1)} Q,$$

where the backward difference vectors are defined recursively [305] as

$$\nabla^l \boldsymbol{\pi}_{(m+1)} = \begin{cases} \boldsymbol{\pi}_{(m+1)} & \text{if } l = 0 \\ \nabla^{l-1}\boldsymbol{\pi}_{(m+1)} - \nabla^{l-1}\boldsymbol{\pi}_{(m)} & \text{if } l = 1, \ldots, o \end{cases}. \tag{7.4}$$

BDFo may be rewritten in the form of a linear system of equations as in

$$\boldsymbol{\pi}_{(m+1)} \left(I - \theta_o \Delta t Q\right) = \sum_{l=0}^{o-1} \beta_{o,l} \nabla^l \boldsymbol{\pi}_{(m)}, \tag{7.5}$$

where θ_o, $\beta_{o,0}, \ldots, \beta_{o,o-1} \in \mathbb{R}_{>0}$ are obtained using the coefficients in Table 5.3 of [12] after substituting the expressions for backward difference vectors and rearranging the terms into common parentheses. BDF methods can be used when $o \leqslant 6$. For instance, when $o = 5$ we have $\theta_5 = 60/137$, $\beta_{5,0} = 1$, $\beta_{5,1} = 77/137$, $\beta_{5,2} = 47/137$, $\beta_{5,3} = 27/137$, and $\beta_{5,4} = 12/137$.

The right-hand side in (7.5) can be computed as a linear combination of o backward difference vectors $\nabla^l \boldsymbol{\pi}_{(m)}$ with coefficients $\beta_{o,l}$ from

$$\boldsymbol{\pi}_{(m+1)}^{(rhs)} := \sum_{l=0}^{o-1} \beta_{o,l} \nabla^l \boldsymbol{\pi}_{(m)}. \tag{7.6}$$

Then the linear system

$$\boldsymbol{\pi}_{(m+1)} \left(I - \theta_o \Delta t Q\right) = \boldsymbol{\pi}_{(m+1)}^{(rhs)} \tag{7.7}$$

can be solved using an iterative method when Q is in Kronecker form.

BDFo local error at step $m+1$ due to truncation can be approximated as

$$\mathbf{r}_{(m+1)} \approx \frac{1}{o+1} \left(\boldsymbol{\pi}_{(m+1)} - \boldsymbol{\pi}_{(m+1)}^{(0)}\right), \tag{7.8}$$

where

$$\boldsymbol{\pi}_{(m+1)}^{(0)} := \sum_{l=0}^{o} \nabla^l \boldsymbol{\pi}_{(m)} \tag{7.9}$$

is the *prediction* vector [291]. Note that at step $m+1$ the prediction vector is the sum of $o+1$ backward difference vectors associated with step m, where $\nabla^0 \boldsymbol{\pi}_{(m)} = \boldsymbol{\pi}_{(m)}$. Hence, the prediction vector at step $m+1$ can be viewed as the approximation obtained in step m plus the correction vector $\sum_{l=1}^{o} \nabla^l \boldsymbol{\pi}_{(m)}$. We will return to the benefit of having such a prediction vector associated with BDFo in the next section.

In order to obtain an estimate of the local error at step $m+1$, we first compute $\|\mathbf{r}_{(m+1)}\|$. If

$$\|\mathbf{r}_{(m+1)}\| < \texttt{stop\_tol}$$

does not hold, then Δt is discarded. A new step size Δt^* is computed from

$$\Delta t^* := \Delta t \left(\frac{\texttt{stop\_tol}}{\|\mathbf{r}_{(m+1)}\|} \right)^{\frac{1}{o+1}}, \tag{7.10}$$

and the computation for step $m+1$ is repeated until an acceptable step size is found [12, 291]. As in RKo, it is possible to multiply `stop_tol` by a safety factor less than but close to 1 so that the step size changes more conservatively [12], or a second tolerance, `stop_tol'`, can be used to lower bound $\|\mathbf{r}_{(m+1)}\|$ and replace `stop_tol` in (7.10) with (`stop_tol` + `stop_tol'`)/2. The latter approach also guards against abrupt decreases in $\|\mathbf{r}_{(m+1)}\|$ which may be indicative of anomalous computational behavior.

Backward differences are quite useful in that they can also be updated easily when the step size changes [291]. Letting B and B^*, respectively, denote matrices with backward difference vectors for old step size Δt and new step size $\Delta t^* \neq \Delta t$ as in

$$B = \left((\nabla^1 \boldsymbol{\pi}_{(m)})^T \ \cdots \ (\nabla^o \boldsymbol{\pi}_{(m)})^T \right),$$
$$B^* = \left((\nabla^{*1} \boldsymbol{\pi}_{(m)})^T \ \cdots \ (\nabla^{*o} \boldsymbol{\pi}_{(m)})^T \right),$$

new backward difference vectors can be obtained [291] from

$$B^* := BR^*U^*,$$

where R^* and U^* are $(o \times o)$ matrices given entrywise as

$$r^*_{l,l'} := \frac{1}{(l+1)!} \prod_{j=0}^{l} (j - (l'+1)(\Delta t^*/\Delta t)),$$
$$u^*_{l,l'} := \frac{1}{(l+1)!} \prod_{j=0}^{l} (j - (l'+1)) \quad \text{for } l, l' = 0, \ldots, o-1.$$

Since BDFo is an o-step method, the first o solutions $\boldsymbol{\pi}_{(0)}, \ldots, \boldsymbol{\pi}_{(o-1)}$ should be available and $O(\Delta t^o)$ accurate to have a global error of $O(\Delta t^o)$ [12, 305]. This implies that in order to obtain the first o solutions, for instance, when $o = 5$, RFK4(5) can be used without the error estimate as an order 5 method.

An approximative technique for transient analysis of stiff MCs using aggregation appears in [35]. The approach is based on classifying states of the MC as fast and slow and eventually aggregating fast states so as to have a smaller non-stiff MC that approximately models the behavior of the original MC. Under certain conditions, the introduced approximation is exact. It will be interesting to see if such an approach can somehow be put to use efficiently in a Kronecker setting.

Parallel implementation of ODE solvers for initial value problems is discussed in [72]. Issues related to parallelization across the method, across time steps, and across unknowns are investigated. The ideas therein may be useful when parallel implementations of ODE solvers for the transient analysis of MCs in Kronecker form are sought.

Implementation of the BDFo method with RFK$o(o+1)$ for the first o steps discussed in the last two subsections and within the `NSolve` package of the APNN toolbox [7, 22] is available at [116]. The particular implementation is able to work with compact vectors and countably infinite state spaces, which are the subjects of the next section.

7.3 Working with Countably Infinite State Spaces

When the Markovian model at hand has a reachable state space $\mathcal{R}$ that is countably infinite, a number of issues pertaining to transient analysis emerge. In this section, we will be addressing these issues on CTMC models with countably infinite $\mathcal{R}$ from a Kronecker-based perspective for systems of stochastic chemical kinetics [117]. The problem (1.2) to be solved in this particular application area is named the *chemical master equation* (CME), which can be challenging even for two-dimensional models [105]. A detailed overview on this subject may be obtained, for instance, from [161]. We start by presenting the model considered in [118] for transient analysis.

7.3.1 A Cascade Model

The cascade model of a system of stochastic chemical kinetics is associated with a biological process in which adjacent genes produce protein that enhances the expression of a succeeding gene [183]. The version considered here has five molecules each corresponding to a submodel with the transition classes in Table 7.1 for which $H = H_I = 5$, $H_F = 0$, $\mathbf{i} = (i_1, i_2, i_3, i_4, i_5)$, $K = 10$, and a, b, c, $\mu \in \mathbb{R}_{>0}$. We let $a = 0.7$, $b = 1$, $c = 5$, and $\mu = 0.07$ as in [183] for the initial state $\mathbf{i}_{(0)} = (10, 10, 10, 10, 10)$ and final time $t = 10$. Observe in this model that the submodel state spaces are countably infinite, that is, $\mathcal{S}^{(h)} = \mathbb{Z}_{\geqslant 0}$ for $h = 1, \ldots, H$, and $\mathcal{R} = \times_{h=1}^{5} \mathcal{S}^{(h)}$.

A Kronecker representation for the cascade model, which has separable transition rate functions, can be obtained by letting the transition matrix of submodel h for $h = 1, \ldots, H$ and transition class $k = 1, \ldots, K$ be denoted by $Q_k^{(h)} \in \mathbb{R}_{\geqslant 0}^{|\mathcal{S}^{(h)}| \times |\mathcal{S}^{(h)}|}$ and given entrywise as

$$Q_k^{(h)}(i_h, j_h) = \begin{cases} \alpha_k^{(h)}(i_h) & \text{if } j_h = i_h + v_h^{(k)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i_h, j_h \in \mathcal{S}^{(h)} \,.$$

Table 7.1 Transition classes of the cascade model

k	ϕ_k	$\alpha_k^{(1)}(i_1)$	$\alpha_k^{(2)}(i_2)$	$\alpha_k^{(3)}(i_3)$	$\alpha_k^{(4)}(i_4)$	$\alpha_k^{(5)}(i_5)$	$\mathbf{v}^{(k)}$
1	a	1	1	1	1	1	$\mathbf{e}_1^T$
2	μ	i_1	1	1	1	1	$-\mathbf{e}_1^T$
3	b	$\frac{i_1}{bi_1+c}$	1	1	1	1	$\mathbf{e}_2^T$
4	μ	1	i_2	1	1	1	$-\mathbf{e}_2^T$
5	b	1	$\frac{i_2}{bi_2+c}$	1	1	1	$\mathbf{e}_3^T$
6	μ	1	1	i_3	1	1	$-\mathbf{e}_3^T$
7	b	1	1	$\frac{i_3}{bi_3+c}$	1	1	$\mathbf{e}_4^T$
8	μ	1	1	1	i_4	1	$-\mathbf{e}_4^T$
9	b	1	1	1	$\frac{i_4}{bi_4+c}$	1	$\mathbf{e}_5^T$
10	μ	1	1	1	1	i_5	$-\mathbf{e}_5^T$

Then

$$Q = Q_O + Q_D, \quad Q_O = \sum_{k=1}^{K} \phi_k \bigotimes_{h=1}^{H} Q_k^{(h)}, \quad Q_D = -\sum_{k=1}^{K} \phi_k \bigotimes_{h=1}^{H} \mathrm{diag}(Q_k^{(h)}\mathbf{e}).$$

Let us now state the issues that emerge in the transient analysis of countably infinite CTMCs as in the cascade model. First, it is nontrivial to truncate $\mathcal{R}$ in such a way that the number of states in the finite truncation is tractable during transient analysis while the probability of the model being outside the truncated state space remains small. Second, since transition rates are almost always dependent on state values, stiffness manifests itself in the underlying model, and rates become unbounded as state values increase. Consequently, uniformization and RK methods do not seem as viable solution alternatives in this application area.

In the next subsection, we look into how Kronecker-based BDF as discussed in Section 7.2 can be used with JOR to solve the linear system in (7.7) and employ compact solution vectors as discussed in Section 6.6 when the system of ODEs underlying a CTMC are countably infinite.

7.3.2 State Space Truncation and Compact Solution Vectors

Initially the model will be in a finite subset of $\mathcal{R}$, many times in a single state. At time step t_{m+1}, $\mathcal{R}$ may be truncated to give $\mathcal{R}_{(m+1)} \subset \mathcal{R}$ such that $|\mathcal{R}_{(m+1)}| < \infty$ by assuming that the probability of the model being in $\mathcal{R} \backslash \mathcal{R}_{(m+1)}$, that is, in states outside the truncated and finite reachable state space $\mathcal{R}_{(m+1)}$, is small. One possible approach is to gradually expand the truncated state space by adding states that are reachable within a certain number of transitions as in the *finite state projection* (FSP) algorithm [247]

until it includes most of the probability mass and then solving the truncated model with a prespecified tolerance. The FSP algorithm uses the number of dimensions, H, in the model in limiting reachability during the truncated reachable state space expansion process. Unfortunately, this approach yields a truncated state space including many states with negligible probability. More information on the FSP algorithm and its variants can be obtained, for instance, from the survey in [128]. Here, we follow a different approach [117].

The prediction vector $\boldsymbol{\pi}^{(0)}_{(m+1)}$ of BDFo in (7.9) is a good estimate of the solution at time step $t_{m+1} (= t_m + \Delta t)$. Therefore, the truncated reachable state space $\mathcal{R}_{(m+1)}$ will capture most of the probability mass at time t_{m+1} if

$$\sum_{\mathbf{i} \in \mathcal{R} \setminus \mathcal{R}_{(m+1)}} \boldsymbol{\pi}^{(0)}_{(m+1)}(\mathbf{i})$$

is small. This approach allows us to choose $\mathcal{R}_{(m+1)}$ *before* solving (7.5) since the backward difference vectors $\nabla^0 \boldsymbol{\pi}_{(m)}, \ldots, \nabla^o \boldsymbol{\pi}_{(m)}$ forming $\boldsymbol{\pi}^{(0)}_{(m+1)}$ in (7.9) are obtained at the previous time step t_m. Note that it is also relatively easy to compute an aggregated version $\boldsymbol{\pi}^{(0)}_{(m+1,h)}$ of the prediction vector $\boldsymbol{\pi}^{(0)}_{(m+1)}$ for submodel h using the aggregation operator $R^{(h)}$ in (6.24) as

$$\boldsymbol{\pi}^{(0)}_{(m+1,h)} := \boldsymbol{\pi}^{(0)}_{(m+1)} R^{(h)} \quad \text{for } h = 1, \ldots, H. \tag{7.11}$$

Here, we let the truncated reachable state space $\mathcal{R}_{(m+1)}$ at time step t_{m+1} be a Cartesian product of consecutive integers as in the sliding window approach [327]. It will be interesting to consider a tighter and smaller $\mathcal{R}_{(m+1)}$ based on a union of Cartesian products as future work.

At time step t_{m+1}, first the aggregated prediction vector $\boldsymbol{\pi}^{(0)}_{(m+1,h)}$ for submodel h in (7.11) is computed for $h = 1, \ldots, H$. When a compact vector in HTD format is used to represent $(\boldsymbol{\pi}^{(0)}_{(m+1)})^T$ with orthogonal basis matrices $U^{(h)}$ and transfer matrices $B^{(\bar{t})}$ forming $\mathbf{c}$ (see Section 5.3), then the aggregated prediction vector for submodel h follows from (6.24) as

$$\begin{aligned}
(\boldsymbol{\pi}^{(0)}_{(m+1,h)})^T &= \left(\bigotimes_{l=1}^{h-1} \mathbf{e}^T I_{n_l} \right) \otimes I_{n_h} \otimes \left(\bigotimes_{l=h+1}^{H} \mathbf{e}^T I_{n_l} \right) (\boldsymbol{\pi}^{(0)}_{(m+1)})^T \\
&= \left(\bigotimes_{l=1}^{h-1} \mathbf{e}^T U^{(l)} \right) \otimes U^{(h)} \otimes \left(\bigotimes_{l=h+1}^{H} \mathbf{e}^T U^{(l)} \right) \mathbf{c}.
\end{aligned}$$

Observe that the aggregated prediction vector $(\boldsymbol{\pi}^{(0)}_{(m+1,h)})^T$ for submodel h is in HTD format with basis matrices

$$\tilde{U}^{(l)} := \begin{cases} U^{(l)} & \text{if } l = h \\ \mathbf{e}^T U^{(l)} & \text{otherwise} \end{cases}$$

and transfer matrices $B^{(\bar{t})}$ forming $\mathbf{c}$. Then the full-vector representation of $(\boldsymbol{\pi}^{(0)}_{(m+1,h)})^T$ may be obtained by computing the basis matrices at the leaves and moving toward the root in $O(r_{\max}(Hn_{\max} + r^2_{\max} + n_{\max}r_{\max}))$ flops.

Once the aggregated prediction vector for each submodel is computed and a prespecified state truncation tolerance, `state_trunc_tol`, is available, the minimal finite set $\bar{\mathcal{R}}^{(h)}_{(m+1)}$ which is a proper subset of $\mathcal{S}^{(h)}$ satisfying

$$\sum_{i_h \in \mathcal{S}^{(h)} \setminus \bar{\mathcal{R}}^{(h)}_{(m+1)}} \boldsymbol{\pi}^{(0)}_{(m+1,h)}(i_h) < \frac{\texttt{state\_trunc\_tol}}{H}$$

is constructed for $h = 1, \ldots, H$. For submodel h, this condition implies that the states outside $\bar{\mathcal{R}}^{(h)}_{(m+1)}$ have a total aggregated predicted transient probability mass which is upper bounded by $\texttt{state\_trunc\_tol}/H$. We remark that the states in $\bar{\mathcal{R}}^{(h)}_{(m+1)}$ need not be consecutive integers.

Next, using $\bar{\mathcal{R}}^{(h)}_{(m+1)}$, the set

$$\begin{aligned}\mathcal{R}^{(h)}_{(m+1)} := \{&\min(\bar{\mathcal{R}}^{(h)}_{(m+1)}) - \psi|\bar{\mathcal{R}}^{(h)}_{(m+1)}|, \ldots, \\ &\max(\bar{\mathcal{R}}^{(h)}_{(m+1)}) + \psi|\bar{\mathcal{R}}^{(h)}_{(m+1)}|\} \cap \mathcal{S}^{(h)} \quad \text{for } h = 1, \ldots, H,\end{aligned}$$

which consists of consecutive integers encompassing the states in $\bar{\mathcal{R}}^{(h)}_{(m+1)}$ is formed. By construction, this set is also a proper subset of $\mathcal{S}^{(h)}$ and finite. Here, $\psi \in (0, 1)$ is used as a safety factor to enable the expansion of the truncated reachable state space at time step t_{m+1} for each submodel in both directions. Since

$$\bar{\mathcal{R}}^{(h)}_{(m+1)} \subseteq \mathcal{R}^{(h)}_{(m+1)},$$

the transient probability mass that remains outside $\mathcal{R}^{(h)}_{(m+1)}$ is upper bounded by $\texttt{state\_trunc\_tol}/H$, implying that

$$\sum_{\mathbf{i} \in \mathcal{R} \setminus \mathcal{R}_{(m+1)}} \boldsymbol{\pi}^{(0)}_{(m+1)}(\mathbf{i}) < \texttt{state\_trunc\_tol}$$

and $\sum_{\mathbf{i} \in \mathcal{R} \setminus \mathcal{R}_{(m+1)}} \boldsymbol{\pi}^{(0)}_{(m+1)}(\mathbf{i})$ is small when `state_trunc_tol` is small.

Having constructed

$$\mathcal{R}_{(m+1)} := \bigtimes_{h=1}^{H} \mathcal{R}^{(h)}_{(m+1)}$$

as a Cartesian product of consecutive integers, the next thing to do is to obtain the backward difference vectors $\nabla^0 \boldsymbol{\pi}_{(m)}, \ldots, \nabla^o \boldsymbol{\pi}_{(m)}$ and the prediction vector $\boldsymbol{\pi}^{(0)}_{(m+1)}$ incident on $\mathcal{R}_{(m+1)}$. Note that the entries of these vectors obtained at the previous time step t_m correspond to the states of the truncated

reachable state space $\mathcal{R}_{(m)}$ and need to be updated. This can be achieved by setting the entries corresponding to (new) states in $\mathcal{R}_{(m+1)} \backslash \mathcal{R}_{(m)}$ to 0. Other entries corresponding to (old) states in $\mathcal{R}_{(m+1)} \cap \mathcal{R}_{(m)}$ retain their values. When backward difference vectors and the prediction vector are represented in HTD format, rows of the new orthogonal basis and transfer matrices are either zero or can be obtained by copying the corresponding rows from the old orthogonal basis and transfer matrices. Hence, the updating of these vectors requires no flops.

Now, it is possible to move to the solution phase in which JOR(ω) is used to solve (7.7) as in

$$\begin{aligned}
\boldsymbol{\pi}_{(0,m+1)} &:= \boldsymbol{\pi}^{(0)}_{(m+1)}; \\
\boldsymbol{\pi}_{(it+1,m+1)} &:= (1-\omega)\boldsymbol{\pi}_{(it+1,m+1)} \\
&\quad +\omega \left(\boldsymbol{\pi}^{(rhs)}_{(m+1)} + \boldsymbol{\pi}_{(it,m+1)} Q_O(\mathcal{R}_{(m+1)}, \mathcal{R}_{(m+1)}) \right) \star \mathbf{dinv}_{(m+1)} \\
&\qquad\qquad \text{for } it = 0, \ldots, \texttt{jor_maxit} - 1; \\
\boldsymbol{\pi}_{(m+1)} &:= \boldsymbol{\pi}_{(it+1,m+1)},
\end{aligned}$$

where $\boldsymbol{\pi}^{(rhs)}_{(m+1)}$ is computed from (7.6) and the elementwise reciprocated vector $\mathbf{dinv}_{(m+1)}$ corresponding to the vector

$$\mathbf{d}_{(m+1)} := \left(\mathbf{e} - \theta_o \Delta t Q_D(\mathcal{R}_{(m+1)}, \mathcal{R}_{(m+1)}) \mathbf{e} \right)^T$$

is computed using Algorithm 10. In the particular implementation, JOR(ω) stops if the 2-norm of the residual vector

$$nrm_{(it+1,m+1)} := \| \boldsymbol{\pi}^{(rhs)}_{(m+1)} - \boldsymbol{\pi}_{(it+1,m+1)} \left(I - \theta_o \Delta t Q(\mathcal{R}_{(m+1)}, \mathcal{R}_{(m+1)}) \right) \|_2$$

is less than `jor_tol`, `jor_maxit` iterations are performed, or stagnation is detected using *nrm_chg* with `jor_chg_tol` similar to Algorithm 10.

After $\boldsymbol{\pi}_{(m+1)}$ is obtained, $\|\mathbf{r}_{(m+1)}\|_2$ can be computed from (7.8) and compared with `stop_tol` and `stop_tol`$'$. If `stop_tol`$' < \|\mathbf{r}_{(m+1)}\|_2 <$ `stop_tol`, then the new backward difference vectors $\nabla^0 \boldsymbol{\pi}_{(m+1)}, \ldots, \nabla^o \boldsymbol{\pi}_{(m+1)}$ are computed from (7.4), and BDFo continues with the next time step t_{m+2}. Otherwise, a new step size Δt^* is obtained from (7.10) using (`stop_tol`$' +$ `stop_tol`$)/2$ and BDFo repeats computations for time step t_{m+1}. The sequence of operations associated with BDFo described above will be executed for each new time step until the horizon at time t is met.

As accuracy measures of $\boldsymbol{\pi}_{(M)}$, sum of the local errors in BDFo

$$\sum_{m=0}^{M-1} \| \boldsymbol{\pi}_{(m+1)} - \boldsymbol{\pi}^{(0)}_{(m+1)} \|_2 / (o+1)$$

and sum of the state space truncation errors

$$\sum_{m=0}^{M-1} |\boldsymbol{\pi}_{(m+1)}\mathbf{e} - \boldsymbol{\pi}_{(m)}\mathbf{e}|$$

can be considered.

When an iteration vector is not close to the solution, its accurate computation should not be necessary. Furthermore, the imposed accuracy may introduce additional time and memory requirements. Therefore, as in Section 6.6 and as also suggested in [188], we use adaptive truncation strategies in Newton–Schulz and JOR iterations that allow for the relatively inaccurate computation of iteration vectors.

In adaptive truncation strategies, the rank bound should be increased only when it is necessary. Based on the observation that iterative methods do not converge or converge relatively slowly in case of a small rank bound, the rank bound is increased if the ratio of two consecutive residual norms, nrm_chg, is larger than a prespecified tolerance. In Newton–Schulz, the rank bound is initialized to 1 and increased if $nrm\_chg \geqslant$ `rcp_chg_tol`. In JOR, the rank bound is initialized to the maximum rank of the prediction vector $\boldsymbol{\pi}^{(0)}_{(m+1)}$ and increased if $nrm\_chg \geqslant$ `jor_chg_tol`. In the latter case, the stopping criterion is modified so that JOR assumes stagnation if $nrm\_chg \geqslant$ `jor_chg_tol` at two consecutive iterations.

Truncation of a vector may be inaccurate if rank is determined by a bound instead of `trunc_tol`. Hence, errors may be underestimated if backward difference vectors $\nabla^0\boldsymbol{\pi}_{(m)}, \ldots, \nabla^o\boldsymbol{\pi}_{(m)}$, prediction vector $\boldsymbol{\pi}^{(0)}_{(m+1)}$, right-hand side vector $\boldsymbol{\pi}^{(rhs)}_{(m+1)}$, 2-norm of residual vector $nrm_{(it+1,m+1)}$, and 2-norm of local truncation error $\|\mathbf{r}_{(m+1)}\|_2$ are not computed accurately. Therefore, we truncate these vectors only using `trunc_tol`. In other words, fixed and adaptive rank bounds are considered only when computing elementwise multiplication $\mathbf{d} \star \mathbf{dinv}'$ in Algorithm 10, vector $\mathbf{dinv}$ in Algorithm 10, and iteration vector $\boldsymbol{\pi}_{(it+1,m+1)}$ in JOR. Regarding `trunc_tol`, the same tolerance is used when truncating a vector in fixed and adaptive truncation strategies. We, respectively, choose `rcp_tol`$/\sqrt{2H-3}$ and `jor_tol`$/\sqrt{2H-3}$ as `trunc_tol` in Newton–Schulz and JOR. Other vectors are truncated with `trunc_tol` := `stop_tol`$/\sqrt{2H-3}$ which imposes an accuracy of `stop_tol`.

A time limit of `time_limit` := 1,000 seconds is imposed on execution time, and execution stops at the end of a BDFo step if it has exceeded this limit. The maximum iteration counts of Newton–Schulz and JOR are set as `rcp_maxit` := `jor_maxit` := 100. The initial step size is given by $\Delta t := 10^{-5}$. We experiment with values of `stop_tol` $\in \{10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}\}$. Since backward difference vectors may have an error of `stop_tol`, we impose the same error tolerance in state space truncation and BDFo by letting

$$\texttt{state\_trunc\_tol} := \texttt{stop\_tol} \quad \text{and} \quad \texttt{stop\_tol}' := \texttt{stop\_tol}/5.$$

Tolerances in Newton–Schulz and JOR methods are chosen as

$$\texttt{rcp\_tol} := \texttt{jor\_tol} := \texttt{stop\_tol}/2 \quad \text{and}$$
$$\texttt{rcp\_chg\_tol} := \texttt{jor\_chg\_tol} := 1/2$$

so that these methods are efficient yet relatively accurate. The safety factor is set as $\psi := 1/20$ based on the observation that it is sufficiently small and yields accurate results. The minimum step size is given by $\Delta t_{\min} := 10^{-6}$ based on the assumption that this value is sufficiently small. When updating the step size, we do not let the new step size to be smaller than $\Delta t_{\min}$. Finally, nonzeros less than 10^{-16} are considered as 0.

Numerical experiments are conducted to evaluate the effects of using adaptive truncation strategies with three types of BDF*o* solvers using compact vectors in HTD format. In solver type 1, we use a fixed rank bound in Newton–Schulz and JOR with `max_rank` := 1,000. In solver type 2, we use a fixed rank bound with `max_rank` := 1,000 in JOR but an adaptive truncation strategy in Newton–Schulz. In solver type 3, we use adaptive truncation strategies in Newton–Schulz and JOR.

The particular BDF solver used is BDF5 with RFK4(5) to initialize the first five solutions. The relaxation parameter of JOR is set to $\omega := 1$; hence, Jacobi is used. This BDF solver requires nine vectors of length equal to $\max_m |\mathcal{R}_{(m+1)}|$ when implemented with full vectors. Backward differences need six vectors; diagonal, prediction, and solution need three vectors. The value of $\max_m |\mathcal{R}_{(m+1)}|$ hints at the advantage of using compact vectors over full vectors since full-vector implementations require roughly 9 times this value.

The models considered in the numerical experiments are metabolite synthesis with two metabolites and two enzymes [297], which is mentioned in Section 4.4 and used before as a test case, extended toggle switch [189], lambda phage [183, 188, 292], and cascade [183] which is presented in the previous subsection. In general, the BDF(5) type 1 solver with fixed truncation strategies is not able to compute the solution for lower values of `stop_tol`. We observe that time and memory requirements decrease significantly when adaptive truncation is used with Newton–Schulz as in type 2 and 3 solvers. Furthermore, maximum rank of solution vectors changes smoothly over time when adaptive truncation strategies are used. Number of nonzeros allocated for full vectors in the cascade model is more than 2,000 times that of the compact vectors when BDF(5) is used with adaptive truncation strategies and `stop_tol` $= 10^{-8}$. The results indicate that adaptive truncation strategies avoid increase in memory requirements and execution time when fixed truncation tolerance and fixed rank bound lead to an inefficient solver and are to be recommended.

Chapter 8
Conclusion

Multidimensional Markov chains can be used to model systems that are formed of interacting subsystems. This can be achieved by associating a submodel with each subsystem, identifying the state space of each submodel, and characterizing the transitions in which each submodel participates. The states such a model occupies are often a proper subset of the Cartesian product of its submodel state spaces. This subset forms the reachable state space of the model. In order to represent a multidimensional Markov chain compactly, its reachable state space needs to be expressed as a union of reachable state space partitions, where each partition is a Cartesian product of subsets of submodel state spaces. The intimately related operation for Cartesian product in set theory happens to be the Kronecker product in matrix theory.

The transition matrix associated with the reachable state space of a multidimensional Markov chain is a block matrix in which each block captures the transitions from one reachable state space partition to another. Potentially different types of transitions take place between the states of two reachable state space partitions. Therefore, each block in the transition matrix can be expressed as a sum of Kronecker products of submodel transition submatrices. Here, the concepts of block (matrix), sum (of matrices), Kronecker products, and submatrices from matrix theory correspond, respectively, to the concepts of Cartesian product partitioning of the reachable state space, set of transitions (from the reachable state space partition associated with the row index of the block to the reachable state space partition associated with the column index of the block), Cartesian products, and subsets of submodel state spaces from set theory.

The compact representation of multidimensional Markov chains based on Kronecker products is *exact* as long as the reachable state space is finite. The transition matrix associated with this representation has a rich block structure which is nested and recursive. There are block partitionings in this

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2_8

matrix at as many different levels as there are dimensions in the model and at one more level when there are multiple reachable state space partitions. Preprocessing techniques such as reordering, grouping, and lumping can take advantage of this rich block structure to expedite analysis. When the reachable state space is countably infinite, it normally needs to be truncated for analysis purposes. For steady-state analysis, this truncation can be performed at the outset using a suitably chosen Lyapunov function. For transient analysis, however, the truncation needs to be performed at each time step while moving toward the time at which the solution is desired.

In this Kronecker-based context for multidimensional Markov chains, analysis methods rely on an efficient vector–Kronecker product multiplication algorithm. Block iterative methods based on splittings, projection methods preconditioned with block matrices obtained from splittings, and multilevel methods for steady-state analysis come across as a strong set of solvers which should be integrated to software packages working with Kronecker products. Among these, multilevel methods perform better on a larger number of problems in the literature. As for transient analysis, implicit ordinary differential equation solvers can be recommended for continuous-time Markov chains with transition rates at different time scales. Implementation of these solvers requires intricate programming with dynamically allocated, relatively complex data structures, which needs time, careful testing, and tuning. The major challenge in this process is to develop skills to be able to work with the abstract multidimensional reachable state space utilizing Kronecker products.

In order to explain the introduced concepts further, Markovian models from availability, queueing networks, production lines, communications protocols, and stochastic chemical kinetics are considered. In each case, the transition matrix underlying the respective multidimensional Markov chain is expressed using Kronecker products. Approaches that can be employed to avoid unreachable states and handle countably infinite state spaces are indicated. Advantages of using a decompositional iterative method for a particular kind of availability model to achieve fast convergence, two approximative decompositional iterative methods for closed queueing networks with phase-type service distributions and arbitrary buffer sizes when less accurate results can be tolerated, and a matrix analytic method for systems of stochastic chemical kinetics having countably infinite state spaces are shown. Pointers to software used during this process are given.

Recently, there has been progress in representing solution vectors compactly with Kronecker-based decompositions. Although these representations are not exact, their accuracy can be user controlled. Issues that arise in devising and implementing numerical steady-state and transient analyses solvers for multidimensional Markov chains based on Kronecker products using compact vectors are discussed and compared with their full-vector counterparts.

These initial results are encouraging; they show that the sizes of problems that are solvable on a given platform increases when the compact vector approach is used with adaptive strategies that control accuracy.

Parallel implementation of solvers for multidimensional Markov chains based on Kronecker products emerges as an area that requires investigation.

References

1. Adelson-Velskii, G.M., Landis, E.M.: An algorithm for the organization of information. Sov. Math. Dokl. **3**, 1259–1263 (1962)
2. Ajmone Marsan, M., Balbo, G., Conte, G.: A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. ACM Trans. Comput. Syst. **2**, 93–122 (1984)
3. Ajmone Marsan, M., Donatelli, S., Neri, F.: GSPN models of Markovian multiserver multiqueue system. Perform. Eval. **11**, 227–240 (1990)
4. Akar, N., Sohraby, K.: An invariant subspace approach in M/G/1 and G/M/1 type Markov chains. Comm. Stat. Stoch. Model. **13**, 381–416 (1997)
5. Akyildiz, I.F.: Mean value analysis for blocking queueing networks. IEEE Trans. Softw. Eng. **14**, 418–428 (1988)
6. Aldous, D., Shepp, L.: The least variable phase type distribution is Erlang. Stoch. Model. **3**, 467–473 (1987)
7. APNN–Toolbox. `http://www4.cs.uni-dortmund.de/APNN-TOOLBOX/` (2004). Accessed May 27, 2018
8. Artalejo, J.R.: Accessible bibliography on retrial queues. Math. Comput. Model. **30**, 1–6 (1999)
9. Artalejo, J.R.: Accessible bibliography on retrial queues: Progress in 2000–2009. Math. Comput. Model. **51**, 1071–1081 (2010)
10. Artalejo, J.R., Gómez-Corral, A.: Modelling communication systems with phase type service and retrial times. IEEE Commun. Lett. **11**, 955–957 (2007)
11. Artalejo, J.R., Gómez-Corral, A.: Retrial Queueing Systems: A Computational Approach. Springer, Berlin (2008)
12. Ascher, U.M., Petzold, L.R.: Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. SIAM, Philadelphia (1998)
13. Asmussen, S.: Applied Probability and Queues. 2nd ed., Springer, New York (2003)
14. Baldo, L., Fernandes, L.G., Roisenberg, P., Velho, P., Webber, T.: Parallel PEPS tool performance analysis using stochastic automata networks. In: Danelutto, M., Vanneschi, M., Laforenza, D.(eds.) 10th International European Conference on Parallel Processing, Lecture Notes in Computer Science vol 3149, pp. 214–219. Springer, Berlin (2004)
15. Bao, Y., Bozkurt, I.N., Dayar, T., Sun, X., Trivedi, K.S.: Decompositional analysis of Kronecker structured Markov chains. Electron. Trans. Numer. Anal. **31**, 271–294 (2008)
16. Barker, G.P., Plemmons, R.J.: Convergent iterations for computing stationary distributions of Markov chains. SIAM J. Algebra Discr. **7**, 390–398 (1986)

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2

17. Barker, V.A.: Numerical solution of sparse singular linear systems of equations arising from ergodic Markov chains. Comm. Stat. Stoch. Model. **5**, 355–381 (1989)
18. Baumann, H., Dayar, T., Orhan, M.C., Sandmann, W.: On the numerical solution of Kronecker-based infinite level-dependent QBD processes. Technical Report BU-CE-1206, Department of Computer Engineering, Bilkent University, Ankara (2012)
19. Baumann, H., Dayar, T., Orhan, M.C., Sandmann, W.: On the numerical solution of Kronecker-based infinite level-dependent QBD processes. Perform. Eval. **70**, 663–681 (2013)
20. Baumann, H., Sandmann, W.: Numerical solution of level dependent quasi-birth-and-death processes. In: International Conference on Computational Science, Procedia Computer Science, vol 1, pp. 1555–1563. Elsevier, Amsterdam (2010)
21. Baumann, H., Sandmann, W.: Computing stationary expectations in level-dependent QBD processes. J. of Appl. Probab. **50**, 151–165 (2013)
22. Bause, F., Buchholz, P., Kemper, P.: A toolbox for functional and quantitative analysis of DEDS. In: Puigjaner, R., Savino, N.N., Serra, B. (eds.) Quantitative Evaluation of Computing and Communication Systems, Lecture Notes in Computer Science, vol 1469, pp. 356–359. Springer, Heidelberg (1998)
23. Bell, S.L., Williams, R.J.: Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: asymptotic optimality of a threshold policy. Ann. Appl. Probab. **11**, 608–649 (2001)
24. Benoit, A., Brenner, L., Fernandes, P., Plateau, B.: Aggregation of stochastic automata networks with replicas. Linear Algebr. Appl. **386** 111–136 (2004)
25. Benoit, A., Brenner, L., Fernandes, P., Plateau, B., Stewart, W.J.: The Peps software tool. In: Kemper, P., Sanders, W.H. (eds.) Computer Performance Evaluation: Modelling Techniques and Tools, Lecture Notes in Computer Science, vol 2794, pp. 98–115. Springer, Heidelberg (2003)
26. Benoit, A., Fernandes, P., Plateau, B., Stewart, W.J.: On the benefits of using functional transitions and Kronecker algebra. Perform. Eval. **58** 367–390 (2004)
27. Benoit, A., Plateau, B., Stewart, W.J.: Memory-efficient Kronecker algorithms with applications to the modelling of parallel systems. Futur. Gener. Comput. Syst. **22** 838–847 (2006)
28. Benzi, M: Preconditioning techniques for large linear systems: A survey. J. Comput. Phys. **182** 418–477 (2002)
29. Benzi, M., Tůma, M.: A parallel solver for large-scale Markov chains. Appl. Numer. Math. **41**, 135–153 (2002)
30. Berman, A., Plemmons, R.J.: Nonnegative Matrices in the Mathematical Sciences. SIAM, Philadelphia (1994)
31. Bharucha-Reid, A.T.: Elements of the Theory of Markov Processes and Their Applications. McGraw-Hill, New York (1960)
32. Bini, D.A., Latouche, G., Meini, B.: Numerical Methods for Structured Markov Chains. Oxford University, Oxford (2005)
33. Bini, D., Meini, B.: On the solution of a nonlinear matrix equation arising in queueing problems. SIAM J. Matrix Anal. Appl. **17**, 906–926 (1996)
34. Bobbio, A., Horváth, A., Telek, M.: The scale factor: a new degree of freedom in phase-type approximation. Perform. Eval. **56**, 121–144 (2004)
35. Bobbio, A., Trivedi, K.S.: An aggregation technique for the transient analysis of stiff Markov chains. IEEE Trans. Comput. **C-35**, 803–814 (1986)
36. Bournez, K., Maler O., Pnueli A.: Orthogonal polyhedra: Representation and computation. In: Vaandrager, F., van Schuppen, J.H. (eds.) Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol 1569, pp. 46–60. Springer, Heidelberg (1999)
37. Brenner, L., Fernandes, P., Fourneau, J.-M., Plateau, B.: Modelling Grid5000 point availability with SAN. Electron. Notes Theor. Comput. Sci. **232**, 165–178 (2009)

38. Brenner, L., Fernandes, P., Plateau, B., Sbeity, I.: PEPS 2007 — Stochastic automata networks software tool. In: Proceedings of the Fourth International Conference on Quantitative Evaluation of Computer Systems and Technologies, pp. 163–164. IEEE Computer Society, Los Alamitos (2007)
39. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial. 2nd ed., SIAM, Philadelphia (2000)
40. Bright, L., Taylor, P.G.: Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes. Stoch. Model. **11**, 497–525 (1995)
41. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30** (1998), 107–117.
42. Bru, R., Pedroche, F., Szyld, D.B.: Additive Schwarz iterations for Markov chains. SIAM J. Matrix Anal. Appl. **27**, 445–458 (2005)
43. Buchholz, P.: A class of hierarchical queueing networks and their analysis. Queueing Syst. **15**, 59–80 (1994)
44. Buchholz, P.: Exact and ordinary lumpability in finite Markov chains. J. Appl. Probab. **31**, 59–75 (1994)
45. Buchholz, P.: Hierarchical Markovian models: symmetries and reduction. Perform. Eval. **22**, 93–110 (1995)
46. Buchholz, P: A Framework for the Hierarchical Analysis of Discrete Event Dynamic Systems. Habilitation Thesis, Faculty of Computer Science, Dortmund University (1996)
47. Buchholz, P.: An aggregation\disaggregation algorithm for stochastic automata networks. Probab. Eng. Inf. Sci. **11**, 229–253 (1997)
48. Buchholz, P.: An adaptive aggregation/disaggregation algorithm for hierarchical Markovian models. Eur. J. Oper. Res. **116**, 545–564 (1999)
49. Buchholz, P.: Hierarchical structuring of superposed GSPNs. IEEE Trans. Softw. Eng. **25**, 166–181 (1999)
50. Buchholz, P.: Structured analysis approaches for large Markov chains. Appl. Numer. Math. **31**, 375–404 (1999)
51. Buchholz, P.: Exact performance equivalence: An equivalence relation for stochastic automata. Theor. Comput. Sci. **215**, 263–287 (1999)
52. Buchholz, P.: Projection methods for the analysis of stochastic automata networks. In: Plateau, B., Stewart, W.J., Silva, M. (eds.) Numerical Solution of Markov Chains, pp. 149–168. Prensas Universitarias de Zaragoza, Zaragoza (1999)
53. Buchholz, P.: Multilevel solutions for structured Markov chains. SIAM J. Matrix Anal. Appl. **22**, 342–357 (2000)
54. Buchholz, P.: Efficient computation of equivalent and reduced representations for stochastic automata. Comput. Syst. Sci. & Eng. **15**, 93–103 (2000)
55. Buchholz, P.: An iterative bounding method for stochastic automata networks. Perform. Eval. **49**, 211–226 (2002)
56. Buchholz, P.: Adaptive decomposition and approximation for the analysis of stochastic Petri nets. Perform. Eval. **56**, 23–52 (2004)
57. Buchholz, P., Ciardo, G., Donatelli, S., Kemper, P.: Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. INFORMS J. Comput. **12**, 203–222 (2000)
58. Buchholz, P., Dayar, T.: Block SOR for Kronecker structured Markovian representations. Linear Algebr. Appl. **386**, 83–109 (2004)
59. Buchholz, P., Dayar, T.: Comparison of multilevel methods for Kronecker structured Markovian representations. Comput. **73**, 349–371 (2004)
60. Buchholz, P., Dayar, T.: Block SOR preconditioned projection methods for Kronecker structured Markovian representations. SIAM J. Sci. Comput. **26**, 1289–1313 (2005)
61. Buchholz, P., Dayar, T.: On the convergence of a class of multilevel methods for large, sparse Markov chains. SIAM J. Matrix Anal. Appl. **29**, 1025–1049 (2007)

62. Buchholz, P., Dayar, T.: A HTD data structure for the analysis of structured Markov chains. http://ls4-www.cs.tudortmund.de/download/buchholz/report.pdf (2017). Accessed May 27, 2018
63. Buchholz, P., Dayar, T.: Efficient transient analysis of a class of compositional fluid stochastic Petri nets. In: Muller, G., Vieira, M. (eds.) Proceedings of the 48th IEEE/IFIP International Conference on Dependable Networks and Systems. IEEE Computer Society, Los Alamitos (2018)
64. Buchholz, P., Dayar, T., Kriege, J., Orhan, M.C.: Compact representation of solution vectors in Kronecker-based Markovian analysis, In: Agha, G., Van Houdt, B. (eds.) Proceedings of the 13th International Conference on Quantitative Evaluation of Systems, Lecture Notes in Computer Science, vol 9826, pp. 260–276. Springer, Heidelberg (2016)
65. Buchholz, P., Dayar T., Kriege, J., Orhan, M.C.: Compact Vector Multiplication software. http://www.cs.bilkent.edu.tr/~tugrul/software.html (2017). Accessed May 27, 2018
66. Buchholz, P., Dayar, T., Kriege, J., Orhan, M.C.: On compact solution vectors in Kronecker-based Markovian analysis. Perform. Eval. **115**, 132–149 (2017)
67. Buchholz, P., Fischer, M., Kemper, P.: Distributed steady state analysis using Kronecker algebra. In: Plateau, P., Stewart, W.J., Silva, M. (eds.) Numerical Solution of Markov Chains, pp. 76–95. Prensas Universitarias de Zaragoza, Zaragoza (1999)
68. Buchholz, P., Kemper, P.: On generating a hierarchy for GSPN analysis. Perform. Eval. Rev. **26**, 5–14 (1998)
69. Buchholz, P., Kemper, P.: Compact representations of probability distributions in the analysis of superposed GSPNs. In: Proceedings of the 9th International Workshop on Petri Nets and Performance Models, pp. 81–90. IEEE, New York (2001)
70. Buchholz, P., Kemper, P.: Kronecker based representations of large Markov chains. In: Haverkort, B., Hermanns, H., Siegle, M. (eds.) Validation of Stochastic Systems, Lecture Notes in Computer Science, vol 2925, pp. 256–295. Springer, Heidelberg (2004)
71. Buchholz, P., Kriege, J., Felko, I.: Input Modeling with Phase-Type Distributions and Markov Models: Theory and Applications. Springer, Cham (2014)
72. Burrage, K.: Parallel and Sequential Methods for Ordinary Differential Equations. Oxford University, Oxford (1995)
73. Campos, J., Donatelli, S., Silva, M.: Structured solution of asynchronously communicating stochastic models. IEEE Trans. Softw. Eng. **25**, 147–165 (1999)
74. Cao, W.-L., Stewart, W.J.: Aggregation/disaggregation methods for nearly uncoupled Markov chains. J. ACM **32**, 702–719 (1985)
75. Chakravarthy, S.R.: (2013). Analysis of MAP/PH/c retrial queue with phase type retrials - simulation approach. In: Dudin, A., Klimenok, V., Tsarenkov, G., Dudin, S. (eds.) Modern Probabilistic Methods for Analysis of Telecommunication Networks, Communications in Computer and Information Science, vol 356, pp. 37–49. Springer, Berlin (2013)
76. Chan, R.H., Ching, W.K.: Circulant preconditioners for stochastic automata networks. Numer. Math. **87**, 35–57 (2000)
77. Chatelin, F., Miranker, W.L.: Acceleration by aggregation of successive approximation methods. Linear Algebr. Appl. **43**, 17–47 (1982)
78. Choi, H., Szyld, D.B.: Application of threshold partitioning of sparse matrices to Markov chains. In: Proceedings of the IEEE International Computer Performance and Dependability Symposium, pp. 158–165 IEEE Computer Society, Los Alamitos (1996)
79. Choi, B.D., Chang, Y., Kim, B.: MAP1,MAP2/M/c retrial queue with guard channels and its application to cellular networks. TOP **7**, 231–248 (1999)
80. Chung, M.-Y., Ciardo, G., Donatelli, S., He, N., Plateau, B., Stewart, W., Sulaiman, E., Yu, J.: A comparison of structural formalisms for modeling large Markov models. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, pp. 196b. IEEE Computer Society, Los Alamitos (2004)

81. Ciardo, G., Jones, R.L., Miner, A.S., Siminiceanu, R.: Logical and stochastic modeling with SMART. In: Kemper, P., Sanders, W.H. (eds.) Computer Performance Evaluation: Modelling Techniques and Tools, Lecture Notes in Computer Science, vol 2794, pp. 78–97. Springer, Heidelberg (2003)
82. Ciardo, G., Miner, A.S.: A data structure for the efficient Kronecker solution of GSPNs. In: Buchholz, P., Silva, M. (eds.) Proceedings of the 8th International Workshop on Petri Nets and Performance Models, pp. 22–31. IEEE Computer Society, Los Alamitos (1999)
83. Clark, G., Gilmore, S., Hillston, J., Thomas, N.: Experiences with the PEPA performance modelling tools. IEE Softw. **146**, 11–19 (1999)
84. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edition. The MIT, Cambridge (2009)
85. Courtois, P.-J.: Decomposability: Queueing and Computer System Applications. Academic, New York (1977)
86. Courtois, P.-J., Semal, P.: Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition. J. ACM **31**, 804–825 (1984)
87. Courtois, P.-J., Semal, P.: Block iterative algorithms for stochastic matrices. Linear Algebr. Appl. **76**, 59–70 (1986)
88. Czekster, R.M., Fernandes, P., Vincent, J.-M., Webber, T.: Split: a flexible and efficient algorithm to vector–descriptor product. In: Glynn, P.W. (ed.) Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools, 83. Nantes, ACM International Conference Proceeding Series (2007)
89. Czekster, R.M., Fernandes, P., Webber, T.: GTAexpress: A software package to handle Kronecker descriptors. In: Proceedings of the Sixth International Conference on Quantitative Evaluation of Computer Systems and Technologies, pp. 281–282. IEEE Computer Society, Los Alamitos (2009)
90. Çınlar, E.: Introduction to Stochastic Processes. Prentice-Hall, Englewood Cliffs (1975)
91. Dao-Thi, T.-H., Fourneau, J.-M.: Stochastic automata networks with master/slave synchronization: Product form and tensor. In: Al-Begain, K., Fiems, D., Horváth, G. (eds.) Proceedings of the 16th International Conference on Analytical and Stochastic Modeling Techniques and Applications, Lecture Notes in Computer Science, vol 5513, pp. 279–293. Springer, Heidelberg (2009)
92. Davio, M.: Kronecker products and shuffle algebra. IEEE Trans. Comput. **C-30**, 116–125 (1981)
93. Davis, T.A.: Direct Methods for Sparse Linear Systems. SIAM, Philadelphia (2006)
94. Davis, T.A., Gilbert, J.R., Larimore, S.I., Ng, E.G.: A column approximate minimum degree ordering algorithm. ACM Trans. Math. Softw. **30**, 353–376 (2004)
95. Davis, T.A., Gilbert, J.R., Larimore, S.I., Ng, E.G.: Algorithm 836: COLAMD, a column approximate minimum degree ordering algorithm. ACM Trans. Math. Softw. **30**, 377–380 (2004)
96. Dayar, T.: State space orderings for Gauss–Seidel in Markov chains revisited. SIAM J. Sci. Comput. **19**, 148–154 (1998)
97. Dayar, T.: Permuting Markov chains to nearly completely decomposable form. Technical Report BU-CEIS-9808, Department of Computer Engineering and Information Science, Bilkent University, Ankara (1998)
98. Dayar, T.: Effects of reordering and lumping in the analysis of discrete-time SANs. In: Gardy, D., Mokkadem, A. (eds.) Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities, pp. 209–220. Birkhauser, Basel (2000)
99. Dayar, T.: On moments of discrete phase-type distributions. Lecture Notes in Computer Science, vol 3670, pp. 51–63. Springer, Heidelberg (2005)

100. Dayar, T.: Analyzing Markov chains based on Kronecker products. In: Langville, A.N., Stewart, W.J. (eds.) MAM 2006: Markov Anniversary Meeting, pp. 279–300. Boson Books, Raleigh (2006)
101. Dayar, T.: Analyzing Markov Chains using Kronecker Products: Theory and Applications. Springer, New York (2012)
102. Dayar, T., Akar, N.: Computing moments of first passage times to a subset of states in Markov chains. SIAM J. Matrix Anal. Appl. **27**, 396–412 (2005)
103. Dayar, T., Hermanns, H., Spieler, D., Wolf, V.: Bounding the equilibrium distribution of Markov population models. Numer. Linear Algebr. Appl. **18**, 931–946 (2011)
104. Dayar, T., Meriç, A.: Kronecker representation and decompositional analysis of closed queueing networks with phase-type service distributions and arbitrary buffer sizes. Ann. Oper. Res. **164**, 193–210 (2008)
105. Dayar, T., Mikeev, L., Wolf, V.: On the numerical analysis of stochastic Lotka–Volterra models. In: Ganzha, M., Paprzycki, M. (eds.) Proceedings of the 2010 International Multiconference on Computer Science and Information Technology (IMCSIT), Wisla, Poland, pp. 289–296. IEEE, Piscataway (2010)
106. Dayar, T., Noyan, G.N.: Steady-state analysis of Google-like stochastic matrices with block iterative methods. Electron. Trans. Numer. Anal. **38**, 69–97 (2011)
107. Dayar, T., Orhan, M.C.: Kronecker-based infinite level-dependent QBDs. J. Appl. Probab. **49**, 1166–1187 (2012)
108. Dayar, T., Orhan M.C.: Cartesian product partitioning of multi-dimensional reachable state spaces. Technical Report BU-CE-1303, Department of Computer Engineering, Bilkent University, Ankara (2013)
109. Dayar, T., Orhan, M.C.: LDQBD solver. Version 3. `http://www.cs.bilkent.edu.tr/~tugrul/software.html` (2013). Accessed May 27, 2018
110. Dayar, T., Orhan, M.C.: On vector–Kronecker product multiplication with rectangular factors. SIAM J. Sci. Comput. **37**, S526–S543 (2015)
111. Dayar, T., Orhan, M.C.: Cartesian product partitioning of multi-dimensional reachable state spaces. Probab. Eng. Inf. Sci. **30**, 413–430 (2016)
112. Dayar, T., Orhan, M.C.: Steady-state analysis of a multiclass MAP/PH/c queue with acyclic PH retrials. J. Appl. Probab. **53**, 1098–1110 (2016)
113. Dayar, T., Orhan, M.C.: CartesianProductPartitioning software. Version 2. `http://www.cs.bilkent.edu.tr/~tugrul/software.html` (2017). Accessed May 27, 2018
114. Dayar, T., Orhan, M.C.: RetrialQueueSolver software. Version 2. `http://www.cs.bilkent.edu.tr/~tugrul/software.html` (2017). Accessed May 27, 2018
115. Dayar T., Orhan, M.C.: VectorKroneckerProductMultiplication software. Version 2. `http://www.cs.bilkent.edu.tr/~tugrul/software.html` (2017). Accessed May 27, 2018
116. Dayar T., Orhan, M.C.: CompactTransientSolver software. `http://www.cs.bilkent.edu.tr/~tugrul/software.html` (2017). Accessed May 27, 2018
117. Dayar, T., Orhan, M.C.: On compact vector formats in the solution of the chemical master equation with backward differentiation. Numer. Linear Algebr. Appl. e2158 (2018)
118. Dayar, T., Orhan, M.C.: A software tool for the compact solution of the chemical master equation. In: German, R., Hielscher, K.S., Krieger, U. (eds.) MMB 2018: Measurement, Modelling and Evaluation of Computing Systems, Lecture Notes in Computer Science, vol 10740, pp. 312–316. Springer, Cham (2018)
119. Dayar, T., Pentakalos, O.I., Stephens, A.B.: Analytical modeling of robotic tape libraries using stochastic automata. Technical Report TR-97-189, Center of Excellence in Space Data & Information Systems, NASA/Goddard Space Flight Center, Greenbelt, Maryland (1997)

120. Dayar, T., Sandmann, W., Spieler, D., Wolf, V.: Infinite level-dependent QBDs and matrix analytic solutions for stochastic chemical kinetics. Adv. Appl. Probab. **43**, 1005–1026 (2011)
121. Dayar, T., Stewart, W.J.: On the effects of using the Grassmann–Taksar–Heyman method in iterative aggregation–disaggregation, SIAM J. Sci. Comput. **17**, 287–303 (1996)
122. Dayar, T., Stewart, W.J.: Quasi lumpability, lower-bounding coupling matrices, and nearly completely decomposable Markov chains. SIAM J. Matrix Anal. Appl. **18**, 482–498 (1997)
123. Dayar, T., Stewart, W.J.: Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains. SIAM J. Sci. Comput. **21**, 1691–1705 (2000)
124. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. **21** 1253–1278 (2000)
125. Demmel, J.W.: Applied Numerical Linear Algebra. SIAM, Philadelphia (1997)
126. Deuflhard, P., Huisinga, W., Jahnke, T., Wulkow, M.: Adaptive discrete Galerkin methods applied to the chemical master equation. SIAM J. Sci. Comput. **30**, 2990–3011 (2008)
127. Dielissen, V.J., Kaldewaij, A.: (1991) Rectangular partition is polynomial in two dimensions but NP-complete in three. Inform. Process. Lett. **38**, 1–6 (1991)
128. Dinh, K.N., Sidje, R.B.: Understanding the finite state projection and related methods for solving the chemical master equation. Phys. Biol. **13**, Article 035003 (2016)
129. Donatelli, S.: Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space. Perform. Eval. **18**, 21–26 (1993)
130. Duff, I.S., Erisman, A.M., Reid, J.K.: Direct Methods for Sparse Matrices. Clarendon, Oxford (1986)
131. Falin, G.I., Templeton, J.G.C.: Retrial queues. Chapman and Hall, London (1977)
132. Fayolle, G., Malyshev, V.A., Menshikov, M.V.: Topics in the Constructive Theory of Countable Markov Chains. Cambridge University, Cambridge (1995)
133. Fehlberg, E.: Low order classical Runge–Kutta formulas with step size control and their application to some heat transfer problems. Comput. **6**, 61–71 (1970)
134. Fernandes, P., O'Kelly, M.E.J., Papadopoulos, C.T., Sales, A.: Analysis of exponential reliable production lines using Kronecker descriptors. Int. J. of Prod. Res. **51**, 4240–4257 (2013)
135. Fernandes, P., Plateau, B.: Triangular solution of linear systems in tensor product format. Perform. Eval. Rev. **28**(4), 30–32 (2001)
136. Fernandes, P., Plateau, B., Stewart, W.J.: Efficient descriptor–vector multiplications in stochastic automata networks. J. ACM **45**, 381–414 (1998)
137. Fernandes, P., Plateau, B., Stewart, W.J.: Optimizing tensor product computations in stochastic automata networks. RAIRO Oper. Res. **32**, 325–351 (1998)
138. Fernandes, P., Sales, A., Zani, L.: PEPS2015 - Stochastic Automata Networks Software Tool. In: Djemame, K., Kavanagh, R., Armstrong, D. (eds.) Proceedings of the 31st UK Performance Engineering Workshop, pp. 9–23. Leeds (2015)
139. Ferrari, L., Sankar P.V., Sklansky, J.: Minimal rectangular partitions of digitized blobs. Comput. Vis. Graph. Image Process. **28**, 58–71 (1984)
140. Fletcher, R.: Conjugate gradient methods for indefinite systems. In: Watson, G.A. (ed.) Proceedings of the Dundee Conference on Numerical Analysis, Lecture Notes in Mathematics, vol 506, pp. 73–89. Springer, Heidelberg (1976)
141. Fourneau, J.-M.: Discrete time stochastic automata networks: using structural properties and stochastic bounds to simplify the SAN. In: Glynn, P.W. (ed.) Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools, 84. Nantes, ACM International Conference Proceeding Series (2007)

142. Fourneau, J.-M.: Product form steady-state distribution for stochastic automata networks with domino synchronizations. In: Thomas, N., Juiz, C. (eds.) Proceedings of the 5th European Performance Engineering Workshop, Lecture Notes in Computer Science, vol 5261, pp. 110–124. Springer, Heidelberg (2008)
143. Fourneau, J.-M.: Collaboration of discrete-time Markov chains: Tensor and product form. Perform Eval. **67**, 779–796 (2010)
144. Fourneau, J.-M., Kloul, L., Pekergin, N., Quessette, F., Véque, V.: Modelling buffer admission mechanisms using stochastic automata networks. Rev. Ann. Télécommun. **49**, 337–349 (1994)
145. Fourneau, J.-M., Maisonniaux, H., Pekergin, N., Véque, V.: Performance evaluation of a buffer policy with stochastic automata networks. In: IFIP Workshop on Modelling and Performance Evaluation of ATM Technology, vol C-15, pp. 433–451. La Martinique, IFIP Transactions North-Holland, Amsterdam (1993)
146. Fourneau, J.-M., Plateau, B., Stewart, W.J.: An algebraic condition for product form in stochastic automata networks without synchronizations. Perform. Eval. **65**, 854–868 (2008)
147. Fourneau, J.-M., Quessette, F.: Graphs and stochastic automata networks. In: Stewart, W.J. (ed.) Computations with Markov Chains, pp. 217–235. Kluwer, Boston (1995)
148. Fox, B.L., Glynn, P.W.: Computing Poisson probabilities. Commun. ACM **31**, 440–445 (1988)
149. Freund, R.W., Nachtigal, N.M.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. Numer. Math. **60**, 315–339 (1991)
150. Gardner, T.S., Cantor, C.R., Collins, J.J.: Construction of a genetic toggle switch in Escherichia coli. Nature **403**, 339–342 (2000)
151. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
152. Gear, C.W.: Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall, Englewood Cliffs (1973)
153. Gelenbe, E., Mitrani, I.: Analysis and Synthesis of Computer Systems. 2nd ed., Imperial College, London (2010)
154. Ghosh, R.: Scalable Stochastic Models for Cloud Services. PhD Thesis, Department of Electrical and Computer Engineering, Duke University, Durham (2012)
155. Gilbert, J.R.: Predicting structure in sparse matrix computations. SIAM J. Matrix Anal. Appl. **15**, 62–79 (1994)
156. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**, 2340–2361 (1977)
157. Glynn, P.W., Zeevi, A: Bounding stationary expectations of Markov processes. In: Markov Processes and Related Topics: A Festschrift for Thomas G. Kurtz, IMS Collections, vol 4, pp. 195–214. Institute of Mathematical Statistics, Beachwood (2008)
158. Golub, G.H., Van Loan, C.F.: Matrix Computations. 4th ed., Johns Hopkins University, Baltimore (2012)
159. Gómez-Corral, A.: A bibliographical guide to the analysis of retrial queues through matrix analytic techniques. Ann. Oper. Res. **141**, 163–191 (2006)
160. Gordon, J.W., Newell, G.F.: Closed queueing systems with exponential servers. Oper. Res. **15**, 252–267 (1967)
161. Goutsias, J., Jenkinson, G.: Markovian dynamics on complex reaction networks. Phys. Rep. **529**, 199–264 (2013)
162. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics: A Foundation for Computer Science. Addison-Wesley, Reading (1989)
163. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**, 2029–2054 (2010)

164. Grassmann, W.K.: Transient solutions in Markovian queueing systems. Comput. Oper. Res. **4**, 47–56 (1977)
165. Grassmann, W.K. (ed.): Computational Probability. Kluwer, Norwell (2000)
166. Grassmann, W.K., Stanford, D.A.: Matrix analytic methods. In: Grassmann, W.K. (ed.) Computational Probability, pp. 153–204. Kluwer, Norwell (2000)
167. Grassmann, W.K., Taksar, M.I., Heyman, D.P.: Regenerative analysis and steady state distributions for Markov chains. Oper. Res. **33**, 1107–1116 (1985)
168. Greenbaum, A.: Iterative Methods for Solving Linear Systems. SIAM, Philadelphia (1997)
169. Gross, D., Miller, D.R.: The randomization technique as a modeling tool and solution procedure for transient Markov processes. Oper. Res. **32**, 343–361 (1984)
170. Gurvich, I., Armony, M., Mandelbaum, A.: Service-level differentiation in call centers with fully flexible servers. Manage. Sci. **54**, 279–294 (2008)
171. Gurvich, I., Whitt, W.: Service-level differentiation in many-server service systems: A solution based on fixed-queue-ratio routing. Oper. Res. **29**, 567–588 (2007)
172. Gusak, O., Dayar, T.: Iterative aggregation–disaggregation versus block Gauss–Seidel on continuous-time stochastic automata networks with unfavorable partitionings. In: Obaidat, M.S., Davoli, F. (eds.) Proceedings of the 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, pp. 617–623. Orlando (2001)
173. Gusak, O., Dayar, T., Fourneau, J.-M.: Stochastic automata networks and near complete decomposability. SIAM J. Matrix Anal. Appl. **23**, 581–599 (2001)
174. Gusak, O., Dayar, T., Fourneau, J.-M.: Lumpable continuous-time stochastic automata networks. Eur. J. Oper. Res. **148**, 436–451 (2003)
175. Gusak, O., Dayar, T., Fourneau, J.-M.: Iterative disaggregation for a class of lumpable discrete-time stochastic automata networks. Perform Eval. **53**, 43–69 (2003)
176. Hackbusch, W.: Multi-Grid Methods and Applications. Springer, Berlin (1985)
177. Hackbusch, W.: Tensor Spaces and Numerical Tensor Calculus. Springer, Heidelberg (2012)
178. Hackbusch, W., Kühn, S.: A new scheme for the Tensor representation. J. Fourier Anal. Appl. **15**, 706–722 (2009)
179. Haddad, S., Moreaux, P.: Asynchronous composition of high-level Petri nets: a quantitative approach. In: Billington, J., Reisig, W. (eds.) Proceedings of the 17th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science, vol 1091, pp. 192–211. Springer, Heidelberg (1996)
180. Harchol-Balter, M.: Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge University, New York (2013)
181. Haverkort, B.R.: Performance of Computer Communication Systems: A Model-Based Approach. Wiley, New York (1998)
182. Haviv, M.: Aggregation/disaggregation methods for computing the stationary distribution of a Markov chain. SIAM J. Numer. Anal. **24**, 952–966 (1987)
183. Hegland, M., Burden, C., Santoso, L., MacNamara, S., Booth, H.: A solver for the stochastic master equation applied to gene regulatory networks. J. Comput. Appl. Math. **205**, 708–724 (2007)
184. Henzinger, T.A., Jobstmann, B,. Wolf, V.: Formalisms for specifying Markovian population models. In: Bournez, O., Potapov, I. (eds.) Proceedings of the 3rd International Workshop on Reachability Problems, Lecture Notes in Computer Science, vol 5797, pp. 3–23. Springer, Berlin (2009)
185. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. 2nd ed., SIAM, Philadelphia (2002)
186. Hillston, J., Kloul, L.: An efficient Kronecker representation for PEPA models. In: de Alfaro, L., Gilmore, S. (eds.) Proceedings of the 1st Process Algebras and Performance Modeling, Probabilistic Methods in Verification Workshop, Lecture Notes in Computer Science, vol 2165, pp. 120–135. Springer, Heidelberg (2001)

187. Horton, G., Leutenegger, S.: A multi-level solution algorithm for steady state Markov chains. Perform. Eval. Rev. **22**(1), 191–200 (1994)
188. Jahnke, T., Huisinga, W.: A dynamical low-rank approach to the chemical master equation. B. Math. Biol. **70**, 2283–2302 (2008)
189. Jahnke, T., Udrescu, T.: Solving chemical master equations by adaptive wavelet compression. J. Comput. Phys. **16**, 5724–5741 (2010)
190. Jain, A., Sahni, S., Palta, J., Dempsey, J.: Partitioning 3D phantoms into homogeneous cuboids. Int. J. Found. Comput. Sci. **14**, 905–931 (2003)
191. Jensen, A.: Markov chains as an aid in the study of Markov processes. Skand. Aktuartidskr. **36**, 87–91 (1953)
192. Kaufman, L.C.: Matrix methods for queueing problems. SIAM J. Sci. Stat. Comput. **4**, 525–552 (1983)
193. Keil, J.M.: Polygon decomposition. In: Sack, J.R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 491–518. Elsevier, Amsterdam (2000)
194. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Springer, New York (1983)
195. Kemper, P.: Numerical analysis of superposed GSPNs. IEEE Trans. Softw. Eng. **22**, 615–628 (1996)
196. Kendall, D.G.: Stochastic processes occurring in the theory of queues and their analysis by the method of imbedded Markov chains. Ann. Math. Statist. **24**, 338–354 (1953)
197. Kim, C.S., Mushko, V., Dudin, A.: Computation of the steady state distribution for multi-server retrial queues with phase type service process. Ann. Oper. Res. **201**, 307–323 (2012)
198. Kim, J., Kim, B.: A survey of retrial queueing systems. Ann. Oper. Res. **247**, 3–36 (2016)
199. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**, 455–500 (2009)
200. Koury, J.R., McAllister, D.F., Stewart, W.J.: Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains. SIAM J. Algebr. Disc. Math. **5**, 164–186 (1984)
201. Kressner, D., Macedo, F.: Low-rank tensor methods for communicating Markov processes. In: Norman, G., Sanders, W. (eds.) Proceedings of the 11th International Conference on Quantitative Evaluation of Systems, Lecture Notes in Computer Science, vol 8657, pp. 25–40, Springer, Heidelberg (2014)
202. Kressner, D., Tobler, C.: `htucker` — A Matlab toolbox for tensors in hierarchical Tucker format. Technical Report 2012–02, Mathematics Institute of Computational Science and Engineering, Lausanne (2012)
203. Kressner, D., Tobler, C.: Algorithm 941: htucker—A Matlab toolbox for tensors in hierarchical Tucker format. ACM Trans. Math. Softw. **40**(3), Article 22, (2014)
204. Krieger, U.R.: Analysis of a loss system with mutual overflow in a Markovian environment. In: Stewart, W.J. (ed.) Numerical Solution of Markov Chains, pp. 303–328. Marcel Dekker, New York (1991)
205. Krieger, U.R.: Numerical solution of large finite Markov chains by algebraic multigrid techniques. In: Stewart, W.J. (ed.) Computations with Markov Chains, pp. 403–424. Kluwer, Boston (1995)
206. Krieger, U.R.: On a two-level multigrid solution method for finite Markov chains. Linear Algebr. Appl. **223–224**, 415–438 (1995)
207. Kumar, M.S., Sohraby, K., Kiseon, K.: (2013). Delay analysis of orderly reattempts in retrial queueing system with phase type retrial time. IEEE Commun. Lett. **17**, 822–825 (2013)
208. Kurtz, T.G.: The relationship between stochastic and deterministic models for chemical reactions. J. Chem. Phys. **57**, 2976–2978 (1972)

209. Kwiatkowska, M., Mehmood, R., Norman, G., Parker, D.: A symbolic out-of-core solution method for Markov models. Electron. Notes Theor. Comput. Sci. **68**, 589–604 (2002)
210. Langville, A.N., Stewart, W.J.: The Kronecker product and stochastic automata networks. J. Comput. Appl. Math. **167**, 429–447 (2004)
211. Langville, A.N., Stewart, W.J.: Testing the nearest Kronecker product preconditioner on Markov chains and stochastic automata networks. INFORMS J. Comput. **16**, 300–315 (2004)
212. Langville, A.N., Stewart, W.J.: A Kronecker product approximate preconditioner for SANs. Numer. Linear Algebr. Appl. **11**, 723–752 (2004)
213. Latouche, G., Ramaswami, V.: A logarithmic reduction algorithm for quasi-birth-and-processes. J. Appl. Probab. **30**, 650–674 (1993)
214. Latouche, G., Ramaswami, V.: Introduction to Matrix Analytic Methods in Stochastic Modeling. SIAM, Philadelphia (1999)
215. Lee, T.L., Li, T.Y., Tsai, C.H.: HOM4PS-2.0: A software package for solving polynomial systems by the polyhedral homotopy continuation method. Comput. **83**, 109–133 (2008)
216. Leskovec, L., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: An approach to modeling networks. J. Mach. Learn. Res. **11**, 985–1042 (2010)
217. Lipshtat, A., Loinger, A., Balaban, N.Q., Biham, O.: Genetic toggle switch without cooperative binding. Phys. Rev. Lett. **96**, 188101 (2006)
218. Loinger, A., Biham, O.: Stochastic simulations of the repressilator circuit. Phys. Rev. E **76**, 051917 (2007)
219. Loinger, A., Lipshtat, A., Balaban, N.Q., Biham, O.: Stochastic simulations of genetic switch systems. Phys. Rev. E **75**, 021904 (2007)
220. Maehara, H.: Note on induced subgraphs of the unit distance graph E^n. Discrete Comput. Geom. **4**, 15–18 (1989)
221. Mandel, J., Sekerka, B.: A local convergence proof for the iterative aggregation method. Linear Algebr. Appl. **51**, 163–172 (1983)
222. Marek, I.: Aggregation methods of computing stationary distributions of Markov processes. In: Albrecht, J., Collatz, L., Hagedorn, P., Velte, W. (eds.) Eigenvalue Problems and Their Numerical Treatment, International Series in Numerical Mathematics, vol 96, pp. 155–169. Birkhäuser, Basel (1991)
223. Marek, I. Iterative aggregation/disaggregation methods for computing some characteristics of Markov chains. II. Fast convergence. Appl. Numer. Math. **45**, 11–28 (2003)
224. Marek, I., Mayer. P.: Iterative aggregation/disaggregation method for computing stationary probability vectors of Markov type operators. Comput. Math. Appl. **31**, 27–40 (1996)
225. Marek, I., Mayer, P.: Convergence analysis of an iterative aggregation/ disaggregation method for computing stationary probability vectors of stochastic matrices. Numer. Linear Algebr. Appl. **5**, 253–274 (1998)
226. Marek, I., Mayer, P.: Iterative aggregation/disaggregation methods for computing some characteristics of Markov chains. In: Margenov, S., Wasniewski, J., Yalamov, P. (eds.) Large-Scale Scientific Computing, Lecture Notes in Computer Science, vol. 2179, pp. 68–80. Springer, Berlin (2001)
227. Marek, I., Mayer, P., Pultarová, I.: Convergence issues in the theory and practice of iterative aggregation/disaggregation methods. Electron. Trans. Numer. Anal. **35**, 185–200 (2009)
228. Marek, I., Pultarová, I.: A note on local and global convergence analysis of iterative aggregation–disaggregation methods. Linear Algebr. Appl. **413**, 327–341 (2006)
229. Marek, I., Szyld, D.B.: Pseudoirreducible and pseudoprimitive bounded operators. Linear Algebr. Appl. **154–156**, 779–791 (1991)

230. Marek, I., Szyld, D.B.: Iterative and semi-iterative methods for computing stationary probability vectors of Markov operators. Math. Comput. **61**, 719–731 (1993)
231. Marek, I., Szyld, D.B.: Local convergence of the (exact and inexact) iterative aggregation method for linear systems and Markov chains. Numer. Math. **69**, 61–82 (1994)
232. Marek, I., Szyld, D.B.: Comparison theorems for the convergence factor of iterative methods for singular matrices. Linear Algebr. Appl. **316**, 67–87 (2000)
233. Marek, I., Szyld, D.B.: Comparison of convergence of general stationary iterative methods for singular matrices. SIAM J. Matrix Anal. Appl. **24**, 68–77 (2002)
234. Marek, I., Szyld, D.B.: Algebraic Schwarz methods for the numerical solution of Markov chains. Linear Algebr. Appl. **386**, 67–81 (2004)
235. Marie, A.R.: An approximate analytical method for general queueing networks. IEEE Trans. Softw. Eng. **5**, 530–538 (1979)
236. Mateescu, M., Wolf, V., Didier, F., Henzinger, T.A.: Fast adaptive uniformisation of the chemical master equation. IET Syst. Biol. **4**, 441–452 (2010)
237. McAllister, D.F., Stewart, G.W., Stewart, W.J.: On a Rayleigh–Ritz refinement technique for nearly uncoupled stochastic matrices. Linear Algebr. Appl. **60**, 1–25 (1984)
238. McQuarrie, D.A.: Stochastic approach to chemical kinetics. J. Appl. Probab. **4**, 413–478 (1967)
239. Meriç, A.: Kronecker Representation and Decompositional Analysis of Closed Queueing Networks with Phase-Type Service Distributions and Arbitrary Buffer Sizes. MS Thesis, Department of Computer Engineering, Bilkent University, Ankara (2007)
240. Meriç, A.: Software for Kronecker Representation and Decompositional Analysis of Closed Queueing Networks with Phase-Type Service Distributions and Arbitrary Buffer Sizes. `http://www.cs.bilkent.edu.tr/~tugrul/software.html` (2007). Accessed May 27, 2018
241. Meyer, C.D.: Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. SIAM Rev. **31**, 240–272 (1989)
242. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia (2000)
243. Migallón, V., Penadés, J., Syzld, D.B.: Block two-stage methods for singular systems and Markov chains. Numer. Linear Algebr. Appl. **3**, 413–426 (1996)
244. Miranker, W.L., Pan, V.Y.: Methods of aggregation. Linear Algebr. Appl. **29**, 231–257 (1980)
245. Mitra, D., Mitrani, I.: Analysis of a Kanban Discipline for Cell Coordination in Production Lines, II: Stochastic Demands. Oper. Res. **39**, 807–823 (1991)
246. Morishima, M., Seton, F.: Aggregation in Leontief matrices and the labour theory of value. Econometrica **29**, 203–220 (1961)
247. Munsky, B., Khammash, M.: The finite state projection algorithm for the solution of the chemical master equation. J. Chem. Phys. **124**, 044104 (2006)
248. Netlib. A collection of mathematical software, papers, and databases. `http://www.netlib.org` (2017). Accessed May 27, 2018
249. Neumann, N., Plemmons, R.J.: Convergent nonnegative matrices and iterative methods for consistent linear systems. Numer. Math. **31**, 265–279 (1978)
250. Neuts, M.F. (1979). A versatile Markovian point process. J. Appl. Probab. **16**, 764–779 (1979)
251. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models: An Algorithhmic Approach. Johns Hopkins University, Baltimore (1981)
252. Neuts, M.F.: Structured Stochastic Matrices of M/G/1 Type and Their Applications. Marcel Dekker, New York (1989)
253. O'Neil, J., Szyld, D.B.: A block ordering method for sparse matrices. SIAM J. Sci. Stat. Comput. **11**, 811–823 (1990)

254. Oppenheim, I., Shuler, K.E., Weiss, G.H.: Stochastic and deterministic formulation of chemical rate equations. J. Chem. Phys. **50**, 460–466 (1969)
255. Orhan, M.C.: Kronecker-based Infinite Level-Dependent QBDs: Matrix Analytic Solution versus Simulation. MS Thesis, Department of Computer Engineering, Bilkent University, Ankara (2011)
256. Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**, 2295–2317 (2011)
257. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. **16**, 973–989 (1987)
258. PEPA Home Page. `http://www.dcs.ed.ac.uk/pepa/tools/` (2005). Accessed May 27, 2018
259. PEPS Home Page. `http://www-id.imag.fr/Logiciels/peps` (2007). Accessed May 27, 2018
260. Pereyra, V., Scherer, G.: Efficient computer manipulation of tensor products with applications to multidimensional approximation. Math. Comput. **27**, 595–605 (1973)
261. Philippe, B., Sidje, R.B.: Transient solutions of Markov processes by Krylov subspaces. Computations with Markov Chains, Proceedings of the Second International Workshop on the Numerical Solution of Markov Chains, pp. 95–119. Kluwer, Boston (1995)
262. Phung-Duc, T., Kawanishi, K.: Performance analysis of call centers with abandonment, retrial and after-call work. Perform. Eval. **80**, 43–62 (2014)
263. Plateau, B.: On the stochastic structure of parallelism and synchronization models for distributed algorithms. Perform. Eval. Rev. **13**(2), 147–154 (1985)
264. Plateau, B., Atif, K.: Stochastic automata network for modeling parallel systems. IEEE Trans. Softw. Eng. **17**, 1093–1108 (1991)
265. Plateau, B., Fourneau, J.-M.: A methodology for solving Markov models of parallel systems. J. Parallel Distrib. Comput. **12**, 370–387 (1991)
266. Plateau, B., Fourneau, J.-M., Lee, K.-H.: PEPS: A package for solving complex Markov models of parallel systems. In: Puigjaner, R., Ptier, D. (eds.) Modeling Techniques and Tools for Computer Performance Evaluation, pp. 291–305. Palma de Mallorca (1988)
267. Plateau, B.D., Tripathi, S.K.: Performance analysis of synchronization for two communicating processes. Perform. Eval. **8**, 305–320 (1988)
268. Pollett, P.K., Taylor, P.G.: On the problem of establishing the existence of stationary distributions for continuous-time Markov chains. Probab. Eng. Inf. Sci. **7**, 529–543 (1993)
269. Pultarová, I.: Local convergence analysis of iterative aggregation–disaggregation methods with polynomial correction. Linear Algebr. Appl. **421**, 122–137 (2007)
270. Pultarová, I.: Necessary and sufficient local convergence condition of one class of iterative aggregation–disaggregation methods. Numer. Linear Algebr. Appl. **15**, 339–354 (2008)
271. Pultarová, I., Marek, I.: Convergence of multi-level iterative aggregation–disaggregation methods. J. Comp. Appl. Math **236**, 354–363 (2011)
272. Pultarová, I., Marek, I.: Physiology and pathology of iterative aggregation–disaggregation methods. Numer. Linear Algebr. Appl. **18**, 1051–1065 (2011)
273. Ramaswami, V., Lucantoni, D.M.: Algorithm for the multi-server queue with phase-type service. Stoch. Model. **1**, 393–417 (1985)
274. Ramaswami, V., Taylor, P.G.: Some properties of the rate operators in level dependent quasi-birth-and-death processes with a countable number of phases. Stoch. Model. **12**, 143–164 (1996)
275. Reibman, A., Trivedi, K.: Numerical transient analysis of Markov models. Comput. Oper. Res. **15**, 19–36 (1988)

276. Ruge, J.W., Stüben, K.: Algebraic multigrid. In: McCormick, S.F. (ed.) Multigrid Methods, Frontiers in Applied Mathematics 3, pp. 73–130. SIAM, Philadelphia (1987)
277. Saad, Y.: SPARSKIT: A Basic Tool Kit for Sparse Matrix Computation. CSRD Report No. 1029, Center for Supercomputing Research and Development, University of Illinois, Urbana (1990)
278. Saad, Y.: Projection methods for the numerical solution of Markov chain models. In: Stewart, W.J. (ed.) Numerical Solution of Markov Chains, pp. 455–471. Marcel Dekker, New York (1991)
279. Saad, Y.: Preconditioned Krylov subspace methods for the numerical solution of Markov chains. In: Stewart, W.J. (ed.) Computations with Markov Chains, Proceedings of the Second International Workshop on the Numerical Solution of Markov Chains, pp. 49–64. Kluwer, Boston (1995)
280. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia (2003)
281. Saad, Y., Schultz, M.H.: GMRES: A generalized minimum residual algorithm for nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
282. Sakurai, H., Phung-Duc, T.: Two-way communication retrial queues with multiple types of outgoing calls. TOP **23**, 466–492 (2015)
283. Sandmann, W., Wolf, V.: Computational probability for systems biology. In: Fisher, J. (ed.), Formal Methods in Systems Biology, Lecture Notes in Computer Science vol 5054, pp.33–47. Springer, Berlin (2008)
284. Sbeity, I., Brenner, L., Plateau, B., Stewart, W.J.: Phase-type distributions in stochastic automata networks. Eur. J. Oper. Res. **186**, 1008–1028 (2008)
285. Sbeity, I., Plateau, B.: Structured stochastic modeling and performance analysis of a multiprocessor system. In: Langville, A.N., Stewart, W.J. (eds.) MAM 2006: Markov Anniversary Meeting, pp. 301–314. Boson Books, Raleigh (2006)
286. Scarpa, M., Bobbio, A.: Kronecker representation of stochastic Petri nets with discrete PH distributions. In: Proceedings of the IEEE International Computer Performance and Dependability Symposium, pp. 52–61. IEEE Computer Society, Los Alamitos (1998)
287. Schuster J., Siegle, M.: A multilevel algorithm based on binary decision diagrams. In: Al-Begain, K., Heindl, A., Telek, E. (eds.) Proceedings of the 14th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA'07), pp. 129–136. Prague (2007)
288. Schuster, J.: Towards Faster Numerical Solution of Continuous Time Markov Chains Stored by Symbolic Data Structures. PhD Dissertation, Faculty of Computer Science, Universität der Bundeswehr München, Neubiberg (2012)
289. Schweitzer, P.J.: A survey of aggregation–disaggregation in large Markov chains. In: Stewart, W.J. (ed.) Numerical Solution of Markov Chains, pp. 53–80, Marcel Dekker, New York (1991)
290. Seneta, E.: Non-negative Matrices: An Introduction to Theory and Applications. Allen & Unwin, London (1973)
291. Shampine, L.F., Reichelt, M.W.: The MATLAB ODE Suite. SIAM J. Sci. Comput. **18**, 1–22 (1997)
292. Shea, M.A., Ackers, G.K.: The O_R control system of bacteriophage lambda: A physical-chemical model for gene regulation. J. Mol. Biol. **181**, 211–230 (1985)
293. Sheskin, T.J.: A Markov partitioning algorithm for computing steady-state probabilities. Oper. Res. **33**, 228–235 (1985)
294. Sidje, R.B., Stewart, W.J.: A numerical study of large sparse matrix exponentials arising in Markov chains. Comput. Stat. Data Anal. **29** 345–368 (1999)
295. Simon, H.A., Ando, A.: Aggregation of variables in dynamic systems. Econometrica **29**, 111–138 (1961)

296. Singer, K.: Application of the theory of stochastic processes to the study of irreproducible chemical reactions and nucleation processes. J. Roy. Statist. Soc. Ser. B **15**, 92–106 (1953)
297. Sjöberg, P.L., Lötstedt, P., Elf, J.: Fokker–Planck approximation of the master equation in molecular biology, Comput. Vis. Sci. **12**, 37–50 (2009)
298. Soltan, V., Gorpinevich, A.: (1993) Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. Discrete Comput. Geom. **9**, 57–79 (1993)
299. Sonneveld, P.: CGS: A fast Lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **10**, 36–52 (1989)
300. Staff, P.J.: (1970). A stochastic development of the reversible Michaelis–Menten mechanism. J. Theor. Biol. **27**, 221–232 (1970)
301. Stepanov, S.N.: Markov models with retrials: The calculation of stationary performance measures based on concept of truncation. Math. Comput. Model. **30**, 207–228 (1999)
302. Stewart, G.W.: Matrix Algorithms, Vol I: Basic Decompositions. SIAM, Philadelphia (1998)
303. Stewart, G.W.: Matrix Algorithms, Vol II: Eigensystems. SIAM, Philadelphia (2002)
304. Stewart, G.W., Stewart, W.J., McAllister, D.F.: A two-stage iteration for solving nearly completely decomposable Markov chains. In: Golub, G.H., Greenbaum, A., Luskin, M. (eds.) The IMA Volumes in Mathematics and its Applications 60: Recent Advances in Iterative Methods, pp. 201–216. Springer, New York (1994)
305. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University, Princeton (1994)
306. Stewart, W.J.: Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton University, Princeton (2009)
307. Stewart, W.J., Atif, K., Plateau, B.: The numerical solution of stochastic automata networks. Eur. J. Oper. Res. **86**, 503–525 (1995)
308. Stewart, W.J., Wu, W.: Numerical experiments with iteration and aggregation for Markov chains. ORSA J. Comput. **4**, 336–350 (1992)
309. Takahashi, Y.: A lumping method for numerical calculations of stationary distributions of Markov chains. Research Report B-18, Department of Information Sciences, Tokyo Institute of Technology, Tokyo (1975)
310. Tewarson, R.P.: Sparse Matrices. Academic, New York (1973)
311. Thattai, M., van Oudenaarden, A.: Intrinsic noise in gene regulatory networks. Proc. Natl. Acad. Sci. **98**, 8614–8619 (2001)
312. Touzene, A.: A tensor sum preconditioner for stochastic automata networks. INFORMS J. Comput. **20**, 234–242 (2008)
313. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika **3**, 279–311 (1966)
314. Tweedie, R.L.: Sufficient conditions for regularity, recurrence and ergodicity of Markov processes. Math. Proc. Camb. Philos. Soc. **78**, 125–136 (1975)
315. Uysal, E., Dayar, T.: Iterative methods based on splittings for stochastic automata networks. Eur. J. Oper. Res. **110**, 166–186 (1998)
316. Vakhutinsky, I.Y., Dudkin, L.M., Ryvkin, A.A.: Iterative aggregation — A new approach to the solution of large scale problems. Econometrica **47**, 821–841 (1979)
317. van der Vorst, H.A.: BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **13**, 631–644 (1992)
318. van Kampen, N.G.: Stochastic Processes in Physics and Chemistry. Elsevier, Amsterdam (1992)
319. Van Loan, C.F.: Introduction to Scientific Computing: A Matrix–Vector Approach Using MATLAB. Prentice-Hall, Upper Saddle River (2000)
320. Van Loan, C.F.: The ubiquitous Kronecker product. J. Comput. Appl. Math. **123**, 85–100 (2000)

321. Vantilborgh, H.: Aggregation with an error of $O(\epsilon^2)$. J. ACM **32**, 162–190 (1985)
322. Varga, R.S.: Matrix Iterative Analysis. Springer, Berlin (2000)
323. Vèque, V., Ben-Othman, J.: MRAP: A multiservices resource allocation policy for wireless ATM network. Comput. Netw. ISDN Syst. **29**, 2187–2200 (1998)
324. Wesseling, P.: An Introduction to Multigrid Methods. Wiley, Chichester (1992)
325. Wesseling, P., Sonneveld, P.: Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In: R. Rautmann, R. (ed.) Approximation Methods for Navier–Stokes Problems, Lecture Notes in Mathematics, vol 771, pp. 543–562. Springer, Berlin (1980)
326. Wolf, V.: Modelling of biochemical reactions by stochastic automata networks. Electron. Notes Theor. Comput. Sci. **171**, 197–208 (2007)
327. Wolf, V., Goel, R., Mateescu, M., Henzinger, T.A.: Solving the chemical master equation using sliding windows. BMC Syst. Biol. **4**, 42 (2010)
328. Woodside, C.M., Li, Y.: Performance Petri net analysis of communications protocol software by delay equivalent aggregation. In: Proceedings of the 4th International Workshop on Petri Nets and Performance Models, pp. 64–73. IEEE Computer Society, Los Alamitos (1991)
329. Yao, D.D., Buzacott, J.A.: The exponentialization approach to flexible manufacturing systems models with general processing times. Eur. J. Oper. Res. **24**, 410–416 (1986)
330. Ziegler, G.M.: Lectures on Polytopes. Springer, New York (1995)

Index

T. Dayar, *Kronecker Modeling and Analysis of Multidimensional Markovian Systems*, Springer Series in Operations Research and Financial Engineering, https://doi.org/10.1007/978-3-319-97129-2