BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

APPROXIMATE TECHNIQUES FOR MARKOV MODELS

APROXIMATIVNÍ TECHNIKY PRO MARKOVOVY MODELY

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ROMAN ANDRIUSHCHENKO**
AUTOR PRÁCE

SUPERVISOR **RNDr. MILAN ČEŠKA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Andriushchenko Roman**
Obor: Informační technologie
Téma: **Aproximativní techniky pro Markovovy modely**
Approximate Techniques for Markov Models
Kategorie: Formální verifikace

Pokyny:

1. Seznamte se s existujícími aproximačními technikami (zejména s fast adaptive uniformization a s adaptivní agregací stavového prostoru), které dovolují analyzovat složité Markovovy modely.
2. Navrhněte vylepšení existujících agregačních technik (tj. vhodnější agregační strategii či redukci aproximační chyby).
3. Implementujte techniku fast adaptive uniformization a navržené vylepšení agregačních technik v rámci nástroje PRISM
4. Experimentálně porovnejte jejich efektivitu a praktickou užitečnost na vhodné sadě modelů.

Literatura:

1. A. Abate, L. Brim, M. Ceska, and M. Kwiatkowska. Adaptive Aggregation of Markov Chains: Quantitative Analysis of Chemical Reaction Networks. In *CAV'15*, LNCS, pages 195-213, Springer, 2015.
2. M. Ceska, F. Dannenberg, N. Paoletti, M. Kwiatkowska and L. Brim. Precise Parameter Synthesis for Stochastic Biochemical Systems. In *Acta Informatica*, pages 1-35, Springer, 2016.
3. M. Kwiatkowska, G. Norman and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *CAV'11)*, LNCS, pages 585-591, Springer, 2011.

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání a alespoň začátek práce na bodě třetím.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Češka Milan, RNDr., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

L.S. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta informačních technologií Ústav inteligentních systémů 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstract

In this work we discuss approximative techniques for the analysis of Markov chains, namely, state space aggregation and truncation. First, we focus on the application of the former method for the analysis of discrete-time models: we redesign the clustering algorithm to handle chains with an arbitrary structure of the state space and, most importantly, we improve upon existing bounds on the approximation error. The developed approach is then integrated with uniformisation techniques, in both standard and adaptive forms, to approximate continuous-time models as well as provide estimates of the approximation error. This theoretical framework along with existing truncation-based techniques were implemented within PRISM model checker. Experiments confirm that newly derived bounds provide a several orders of magnitude precision improvement without degrading performance. We show that the resulting aggregating approach can provide a valid model approximation supplied by adequate approximation error estimates, in both discrete and continuous time. Then, we perform a comparative analysis of aggregating and truncating techniques, illustrate how different methods handle various types of models, and identify chains for which aggregating, or truncating, analysis is preferred. Finally, we demonstrate a successful usage of approximative techniques for model checking Markov chains.

Abstrakt

Předkládaná práce je zaměřena na popis aproximativních technik pro analýzu Markovských řetězců, konkrétně na metody založené na agregaci nebo ořezávání stavového prostoru. Na začátku je představen postup umožňující aplikaci agregace pro modely diskrétního času s libovolnou strukturou stavového prostoru a je odvozen lepší odhad aproximační chyby. Daný postup je pak propojen s uniformizačními technikami, jak se standardní tak s adaptivní, což umožňuje provádět analýzu řetězců spojitého času spolu s odhadem aproximační chyby. Navržená technika spolu s existujícími metodami založenými na ořezávání byly implementovány v rámci nástroje PRISM. Provedené experimenty potvrzuji, že nově odvozený odhad aproximační chyby vylepšuje přesnost o několik řádů bez zhoršení celkové výkonnosti. Je ukázano, že výsledná agregační metoda je schopna poskytnout validní aproximaci modelu spolu s adekvátními odhady aproximační chyby, a to jak v diskrétním tak i ve spojitém čase. Následně je provedeno porovnání s technikami založenými na ořezávání stavového prostoru a je diskutováno pro které třídy Markovských řetězců je ta či ona metoda použitelnější. Nakonec je demonstrováno úspěšne použití aproximativních technik pro model checking Markovových modelů.

Keywords

Markov models, probabilistic model checking, approximation techniques, adaptive aggregation

Klíčová slova

Markovovy modely, probabilistický model checking, aproximativní techniky, adaptivní agregace

Reference

ANDRIUSHCHENKO, Roman. *Approximate Techniques for Markov Models.* Brno, 2018. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor RNDr. Milan Češka, Ph.D.

Approximate Techniques for Markov Models

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of RNDr. Milan Češka, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Roman Andriushchenko
May 14, 2018

Acknowledgements

I would like to thank my supervisor RNDr. Milan Češka, Ph.D. for his encouragement and continuous support throughout the work on this project.

Contents

Chapter 1

Introduction

Probability plays a prominent role in the design and modelling of systems with unpredictable or unreliable behaviour. Markov chains are a class of such probabilistic modelling tools that have been extensively used in many areas of science and engineering, including analysis of performance of computer networks, reliability of communication and security protocols [2, 4], in the study of various quantitative attributes of biochemical reaction networks [19, 5] or genetics [14]. A Markov chain can be thought of as a collection of states accompanied by function that describes a probabilistic nature of a transition between any pair of states. Depending on the type of the model, these transitions can occur in discrete or continuous time. An analysis of such chain is carried out through simulation-based exploration of its execution paths or using numerical schemes, usually by solving a system of equations. For an analysis of continuous-time models, a typical method employed is uniformisation, which is based on a time-discretisation of the chain of interest [23].

Unfortunately, an efficient analysis Markov models is difficult to achieve in practice due to the state-space explosion problem. In order to enable the handling of larger state spaces, several approximation techniques have been introduced. These techniques typically solve a smaller chain – the one with the reduced state space – and then interpret results in terms of the original model. State aggregation methods [1] construct this smaller chain by clustering the state space. State-space truncation methods [7], on the other hand, work by dynamically neglecting states with insignificant probability. In both cases an approximation error has to be quantified. In practice, highly accurate probability estimates are crucial, for example in reliability analysis of safety-critical systems or when checking satisfiability of temporal logic formulae.

Key contributions

In this work we expand the existing framework for the analysis of Markov chains via state-space aggregation and provide its first in-depth comparison with truncation-based techniques. Inspired by the application of the adaptive approach developed for the analysis of biochemical reaction networks in [1], we first focus on the design of an accurate and efficient aggregation method applicable to chains with an arbitrary structure of the state space. We start in the discrete setting and redefine a notion of the state-space abstraction in order to arrive at precise bounds on the approximation error. These results are then used to design a new aggregating scheme that preserves all properties of a Markov chain, and we show that this preservation is necessary for integrating it with uniformisation method to enable analysis of continuous-time models. This integration is then carried out, and

explicit bounds on the approximation error are derived. Finally, we introduce adaptivity to our aggregating scheme that allows reducing the required number of computation steps.

A total of eight approximative methods for Markov chain analysis (5 existing and 3 new ones) were implemented in probabilistic model checker PRISM [17] and were also integrated with model checking algorithms. Experiments confirm that newly derived bounds provide a several orders of magnitude precision improvement without degrading performance. We show that the resulting aggregating approach can provide a valid model approximation supplied by adequate approximation error estimates, in both discrete and continuous cases. Then, we perform a comparative analysis of aggregating and truncating techniques, illustrate how different methods handle various types of models and identify chains for which aggregating, or truncating, analysis is preferred. Finally, we demonstrate a successful usage of approximative techniques for model checking Markov chains.

Related work.

Simulation methods are able to analyse a Markov chain – as well as any other stochastic model – by simulating one-time trajectory of the process; collecting the statistics from multiple realisations then allows to estimate the transient probability distribution. A common example of such technique is Gillespie's Stochastic Simulation Algorithm (SSA) [12, 24]. The main disadvantage of SSA is its slow convergence. Moreover, although simulation-based analysis allows to employ adaptivity (e.g. [9]), these methods in general give weak precision guarantees in the form of confidence intervals. Nonetheless, this approach is suitable for situations where highly accurate probability estimates are not required. SSA, along with other simulation-based techniques, is implemented within a COPASI [15] tool developed for the analysis of biochemical network models.

A widely studied numerical method to deal with large state spaces is truncation which works by neglecting states with insignificant probability, computing an underapproximation of the true probability distribution and then using probability loss as an error estimate. In the context of continuous-time chains, a combination of truncation with adaptive uniformisation is a highly celebrated technique known as fast adaptive uniformisation (FAU) [21, 7, 6]. Truncation techniques utilise the fact that usually a significant portion of the probability mass is concentrated in a small subset of the state space and can result in poor accuracy if this mass is spread over a large number of states.

A common representative of the state-space aggregation techniques is a clustering based on (bi-)simulation equivalence [18]. This technique exploits symmetries of a concrete model and performs exact numerical computation but, unfortunately, can only be applied to a specific domain of Markov processes. The work of [8] presents an algorithm to approximate probability distributions of a Markov model forward in time, which served as an inspiration of the adaptive scheme proposed in [1], where a formal error analysis steers the adaptation. This novel use of derived error bounds allows far greater accuracy and flexibility as it accounts also for the past history of the probability mass within specific clusters.

Structure of this paper.

In Chapter 2 we give an overview of the necessary theory regarding both discrete- and continuous-time Markov processes; we describe an aggregation approach presented in [1] and revise uniformisation technique, as well as its fast adaptive version. In Chapter 3 we redefine a notion of a state-space aggregation of a discrete-time Markov chain, introduce techniques that will allow us to approximate chains with arbitrary structure of the state

space, explore various aggregation strategies and, most importantly, we derive a more precise approximation error bound. Lastly, we perform a thorough experimental evaluation of all approximative techniques. In Chapter 4 we develop state-space aggregation for continuous Markov processes by combining results from the previous chapter with both standard and adaptive uniformisation, as well as derive explicit bounds on the approximation error, and discuss experimental results. Finally, in Chapter 5, we give notes on the implementation and collect all the facts and issues that could serve as a departure point for the follow-up research.

Chapter 2

Preliminaries

In this chapter we review the necessary theory and introduce notation that will be used throughout the paper. First, we discuss discrete-time Markov chains along with existing exact and approximate techniques for their analysis. Then we generalise the argument into the continuous case and describe procedures for handling continuous-time models.

2.1 Discrete-Time Markov Chains

Definition 1. [17, 3] *A discrete-time Markov chain (DTMC)* is a pair $D = (S, P)$, where

- S is *the set of states* and
- $P : S \times S \to [0, 1]; \forall r \in S \ \sum_{s \in S} P(r, s) = 1$ is *the transition probability function.*

Set S describes all possible states of the model and expression $\mathbb{P}(D(k) = s), s \in S, k \in \mathbb{N}_0$ denotes the probability that DTMC D resides at state s at time k. The function P establishes probabilities of transitions between the states, namely:

$$\begin{aligned}&\mathbb{P}(D(k+1) = s \mid D(k) = r, D(k-1) = r_{k-1}, \ldots, D(0) = r_0) = \\ &= \mathbb{P}(D(k+1) = s \mid D(k) = r) =: P(r, s),\end{aligned}$$

i.e. transition probability from state r to s applies whenever state r is visited, regardless of what has happened in the past: the next state of the process depends only on its present state. This assumption is known as the *Markov property*. Unless stated otherwise, we will assume that the state space S is finite. A state s for which $P(s, s) = 1$ will be called *absorbing*. Sometimes it is helpful to lay out the model in the so-called *transition probability graph*, whose nodes are the states and whose arcs are non-zero transitions, see Figure 2.1.

The model is initialised via the distribution $p_0(s) \coloneqq \mathbb{P}(D(0) = s)$ and its transient probability distribution $p_k(s) \coloneqq \mathbb{P}(D(k) = s)$ at time step $k > 0$ is

$$p_k(s) = \sum_{r \in S} p_{k-1}(r) P(r, s). \tag{2.1}$$

We denote $\mathbf{p_k} \coloneqq [p_k(s)]_{s \in S}$ to be the row vector of transient probabilities at time k and $\mathbf{P} \coloneqq [P(r, s)]_{r,s \in S}$ to be the transition probability matrix. Recursion (2.1) can be then equivalently expressed as a vector-matrix multiplication:

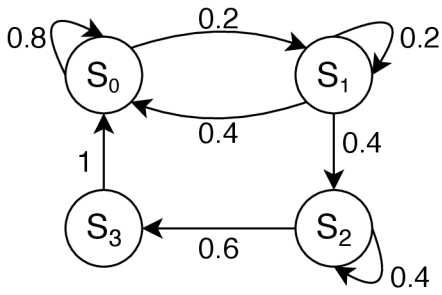

Figure 2.1: A simple DTMC.

$$\mathbf{p_k} = \mathbf{p_{k-1}} \cdot \mathbf{P}. \tag{2.2}$$

An act of performing one such multiplication will be called *an iteration*, *a probability propagation*, *a discrete time-step* or simply *a step*. A problem of finding transient distribution $\mathbf{p_k}$ is often referred to as *the transient analysis* of the chain. Computing this vector directly using (2.2) typically suffers from the state space explosion problem and we are therefore interested in providing an efficient and accurate approximation.

Example 1. Let $D = (S, P)$ be DTMC as in Figure 2.1 that starts at state s_0, i.e. $\mathbf{p_0} = [1, 0, 0, 0]$ and the corresponding transition probability matrix is

$$\mathbf{P} = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0.4 & 0.6 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Let us compute $\mathbf{p_4}$:

$$\begin{aligned} \mathbf{p_1} &= \mathbf{p_0} \cdot \mathbf{P} = [0.8, 0.2, 0, 0]; \\ \mathbf{p_2} &= \mathbf{p_1} \cdot \mathbf{P} = [0.72, 0.2, 0.08, 0]; \\ \mathbf{p_3} &= \mathbf{p_2} \cdot \mathbf{P} = [0.656, 0.184, 0.112, 0.048]; \\ \mathbf{p_4} &= \mathbf{p_3} \cdot \mathbf{P} = [0.6464, 0.168, 0.1184, 0.0672]. \end{aligned}$$

The final distribution describes probabilities of residing in each of the states at time 4, e.g. there is less than 0.7 % probability that DTMC will end up at state s_3.

2.1.1 Model Checking

In general, a stochastic model checking [17] is a method for verifying whether a system exhibits a certain property by calculating the likelihood of occurrence of various events during its execution. Model checking algorithms input a description of a model along with specification expressed in probabilistic temporal logic, and return a probability for a given model to satisfy this property. In the context of DTMCs, as a specification language we use *Probabilistic Computation Tree Logic (PCTL)*, an extension of *Computation Tree Logic*

(CTL). Although expressive capabilities of PCTL are quite rich, the primary goal of this paper is to show that approximation techniques can be efficiently integrated with model checking algorithms. Therefore, we will restrict ourselves only to specific types of formulae, the resulting framework can then be easily generalised to handle any kind of specification. An exhaustive description of PCTL syntax and of model checking procedures is presented in [17]. Here we will define properties of interest and algorithms for their evaluation in a straightforward way.

Let $D = (S, P)$ be DTMC with initial distribution p_0. Let A be a predicate over states in S and let $Sat(A)$ denote the set of states that satisfy A. An expression $[\Diamond^{\leq k} A]$ asserts a property of the model of eventually reaching any of the states in $Sat(A)$ within first k time-steps, assuming initial distribution p_0. Similarly, a formula $[\Box^{\leq k} A]$ represents an event of never leaving subset of states $Sat(A)$ within first k time-steps, assuming initial distribution p_0. Here we assume that the time horizon k is finite and consider only time-bounded specifications. Operator $\Diamond$ is called 'eventually', 'future' or 'diamond'; operator $\Box$ is called 'always', 'globally' or 'box'. By $\mathbb{P}(\Diamond^{\leq k} A)$ or $\mathbb{P}(\Box^{\leq k} A)$ we will denote the likelihood of the corresponding event happening. The following definition will help us compute these probabilities.

Definition 2. Let $D = (S, P)$ be DTMC and let $Sat(A) \subseteq S$. A *PCTL driven transformation of D given A* is a DTMC $D[A] = (S, P[A])$ where

$$P[A](r, s) = \begin{cases} 1, & \text{if } r \in Sat(A) \text{ and } r = s \\ 0, & \text{if } r \in Sat(A) \text{ and } r \neq s \\ P(r, s), & \text{otherwise.} \end{cases}$$

In other words, the resulting chain is this same chain with the states in $Sat(A)$ made absorbing, see Figure 2.2.

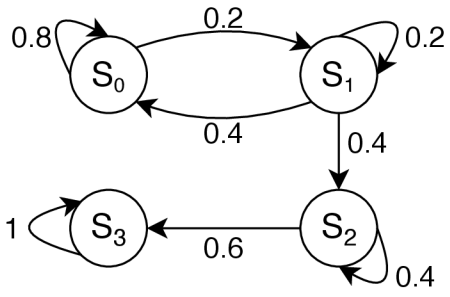

Figure 2.2: A PCTL driven transformation of DTMC from Figure 2.1 given $A \equiv s \in \{s_3\}$.

Proposition 1. Let $D = (S, P)$ be DTMC with initial distribution p_0. Let $D[A]$ be its PCTL driven transformation given A having the same initial distribution $p_0^{D[A]} := p_0$ and denote $p_k^{D[A]}$ to be its transient probability distribution at time k. Then $\mathbb{P}(\Diamond^{\leq k} A) = \sum_{a \in Sat(A)} p_k^{D[A]}(a)$.

Justification of this proposition goes as follows. By making each of the states $a \in Sat(A)$ absorbing, we ensure that any probability mass that reaches a never leaves this state. After k discrete steps, the total probability mass accumulated in a gives a probability of reaching it withing k steps; the sum over all states in A then produces the desired result.

Notice that $\mathbb{P}(\Box^{\leq k} A) = 1 - \mathbb{P}(\Diamond^{\leq k} \neg A)$, i.e. the event of never leaving subset $Sat(A)$ complements the event of eventually reaching a state in $Sat(\neg A) = S \setminus Sat(A)$. Using Proposition 1, we arrive at the result:

$$\mathbb{P}(\Box^{\leq k} A) = 1 - \sum_{a' \in S \setminus Sat(A)} p_k^{D[\neg A]}(a') = \sum_{a \in A} p_k^{D[\neg A]}(a).$$

2.1.2 Adaptive State-Space Aggregation

A high-level description of an aggregation of a Markov model would be a clustering of its state space and then treating resulting clusters as states of a new Markov chain. Defining a suitable transition probability function on this new clustered state space and working within this abstract framework allows us to approximate transient probabilities of a concrete, unaggregated model, as well as provide bounds on the approximation error. A performance increase is achieved since now we are dealing with a smaller model. Our starting point will be the aggregation scheme presented in [1], application of which to discrete-time chains is described in this subsection.

Let $D = (S, P)$ be DTMC. Let $\Phi = \{\varphi_1, ..., \varphi_n\}$, $\bigcup_{i=1}^{n} \varphi_i = S$, $i \neq j \Rightarrow \varphi_i \cap \varphi_j = \emptyset$ form a partition on S and be called *the abstract (aggregated) state space*. Elements $\varphi \in \Phi$ will be called *abstract states* or *clusters*. By expression $|\varphi|$ we will denote the size of the cluster, i.e. the number of concrete states comprising it. Clusters of size 1 will be called *trivial*. Let $\Pi : \Phi \times \Phi \to \mathbb{R}_{\geq 0}$ defined as

$$\Pi(\rho, \sigma) = \frac{1}{|\sigma|} \sum_{r \in \rho} \sum_{s \in \sigma} P(r, s) \tag{2.3}$$

be *the abstract transition probability function*. The intuition behind this equation is that it encompasses the average *incoming* probability to cluster σ from cluster ρ. A pair $\Delta = (\Phi, \Pi)$ describes *the abstract (aggregated) DTMC*. The model is initialised using probability distribution $\pi_0 : \Phi \to \mathbb{R}_{\geq 0}$ defined as

$$\pi_0(\sigma) = \sum_{s \in \sigma} p_0(s),$$

that is, the probability of being in cluster σ at time 0 is the sum of the probabilities of being in either of its concrete states. Transient probability distribution at time step $k > 0$ is then defined recursively:

$$\pi_k(\sigma) = \sum_{\rho \in \Phi} \pi_{k-1}(\rho) \Pi(\rho, \sigma). \tag{2.4}$$

Similarly as before, we denote $\pi_{\mathbf{k}} = [\pi_k(\sigma)]_{\sigma \in \Phi}$ to be the row vector of transient probabilities at time $k \geq 0$, $\mathbf{\Pi} = [\Pi(\rho, \sigma)]_{\rho, \sigma \in \Phi}$ to be the abstract transition probability matrix and express recursion (2.4) as

$$\pi_{\mathbf{k}} = \pi_{\mathbf{k-1}} \cdot \mathbf{\Pi}.$$

Having π_k, we define $\tilde{p}_k : S \to \mathbb{R}_{\geq 0}$ to be an approximation of the concrete transition probabilities $p_k(\cdot)$ and compute it as follows:

$$\tilde{p}_k(s) = \frac{\pi_k(\sigma)}{|\sigma|}, s \in \sigma, \tag{2.5}$$

that is, the probability of a cluster is distributed uniformly between its states. If we pick partition Φ such that $|\Phi| \ll |S|$, working with the abstraction (Φ, Π) will allow us to approximate $\mathbf{p_k}$ using $\pi_\mathbf{k}$ much more easily than performing propagations (2.1) directly. An error associated with this approximation can be derived from the structure of (Φ, Π). Introduce the quantity

$$\epsilon(\rho, \sigma) \coloneqq \max_{s \in \sigma} \left| \Pi(\rho, \sigma) - \frac{|\sigma|}{|\rho|} \sum_{r \in \rho} P(r, s) \right| \tag{2.6}$$

and denote $\epsilon(\rho) \coloneqq \sum_{\sigma \in \Phi} \epsilon(\rho, \sigma)$. Let $e_k(s) \coloneqq \tilde{p}_k(s) - p_k(s)$ be the approximation error for state s at time k and let $\mathbf{e_k} \coloneqq [e_k(s)]_{s \in S}$ denote the corresponding row vector; the overall error is captured by its L_1-norm that is quantified recursilvely:

$$\|\mathbf{e_k}\|_1 \leq \|\mathbf{e_{k-1}}\|_1 + \sum_{\rho \in \Phi} \pi_{k-1}(\rho)\epsilon(\rho), \tag{2.7}$$

where

$$\|\mathbf{e_0}\|_1 = \sum_{s \in S} |p_0(s) - \tilde{p}_0(s)|.$$

The term $\|\mathbf{e_0}\|_1$ is called *aggregation error* and it describes the inaccuracy introduced when we replaced exact p_0 with $\tilde{p}_0$. Additionally, during each discrete step, a *propagation error*, associated with the usage of abstraction Π instead of P, is produced and is captured by $\epsilon(\cdot, \cdot)$: this quantity accounts for the maximum difference, for a given pair of clusters, between the abstract transition probability and (rescaled) pointwise incoming probability. The product of $\pi_{k-1}(\rho)\epsilon(\rho)$ hereby gives the (upper bound of) error generated from ρ, the sum over all abstract states in (2.7) then yields the overall error. The reason for computing the L_1-norm of $\mathbf{e_k}$ and not $\mathbf{e_k}$ itself is that, again, we want to reduce the computation complexity of the error estimation: equation (2.7) suggests that one step of this estimation is equivalent to performing a scalar product of vectors in the abstract (i.e. with reduced state space) setting.

The transformation described above will be referred to as *a state space aggregation based on incoming transition probabilities*. We have not yet discussed how a partition Φ of S is obtained: update equations above can give suggestions on what such partitions should be. First, aggregation error captures point-wise difference $p_0(s) - \tilde{p}_0(s) = p_0(s) - \pi_0(\sigma)/|\sigma| = p_0(s) - \sum_{s' \in \sigma} p_0(s')/|\sigma|$, so we can minimise this error by aggregating together states with similar probability. In Markov chains it is often the case that at any given time instance a majority of the probability mass is concentrated in a particular subset of states and that adjacent states (i.e. those connected by a possible transition) tend to have similar transient probabilities.

Second, a propagation error that depends on both (approximate) transient probability distribution and error factor ϵ suggest that we should minimise $\epsilon(\cdot)$ for clusters that currently have significant transient probability and therefore use clusters of small size for such states. The intuition behind this conclusion is that, when propagating probability from ρ to

σ, we are effectively forwarding probability mass (in one step) to those states in σ that were previously unreachable (in one step) from any of the states in ρ, see Figure 2.3; conversely, we are pushing probability to cluster σ from those states in ρ that do not have any of the states in σ as their direct successor. We are effectively accelerating the system and this probability mass forwarding is the source of the propagation error. Hence, we arrive at a conclusion that the size of the cluster should be inversely proportional to its (approximate) transient probability: a state with significant probability will form a cluster of size 1, and therefore its neighbours, that are more likely to have significant probability as well, are also likely to form a trivial cluster[1]. On the other hand, a group of states with small probability mass can be clustered together: an insignificant $\pi_k()$ will cancel large $\epsilon()$. Finally, states with moderate transient probability will be aggregated to clusters of medium size.

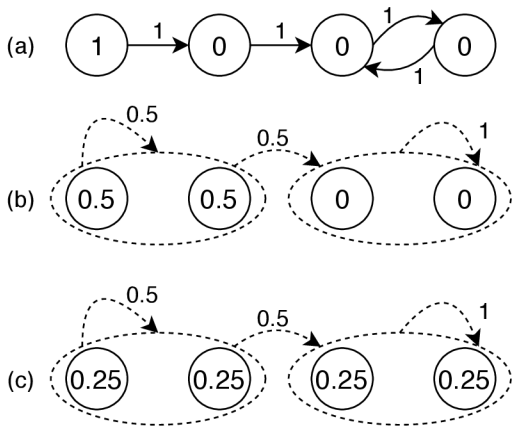

Figure 2.3: In (a) we consider a simple DTMC that deterministically starts in the left-most state (numbers inside nodes denote current transient probabilities); in (b) we construct its aggregation, notice how the first (second) state has effectively lost (gained) some probability mass - this is aggregation error; in (c) we perform one iteration in the abstract setting, observe that the right-most state has effectively gained, due to propagation error, some probability mass: in the unaggregated setting this state is unreachable until after the third iteration.

These two rules gives us a hint, not a recipe, about how a partition is constructed. In [1], a state-space aggregation method was used to analyse biochemical systems and the clustering was created based on the a priori known structure of the model as well as on the knowledge of underlying physical phenomena. Later, in Chapter 3, we will develop a different approach that will help us analyse chains with an arbitrary structure of their state space. Until then, assume that a specific clustering is given.

Having the partition, we can compute $\pi_{\mathbf{0}}$, estimate aggregation error $\|\mathbf{e_0}\|_1$, establish abstract transition probability matrix $\mathbf{\Pi}$ and construct a vector of error factors $\epsilon := [\epsilon(\sigma)]_{\sigma\in\Phi}$. We then proceed by propagating probability mass using abstract structures and quantify error using update equation (2.7). The final result $\tilde{\mathbf{p}}_{\mathbf{k}}$ is then constructed by deaggregating the state space and the probability distribution $\pi_{\mathbf{k}}$, as in (2.5).

Also, as we perform discrete steps, a probability distribution changes and at some time a cluster with large $\epsilon(\cdot)$ may accumulate a lot of probability mass and start to produce a significant error. Hence, we need to *adapt* our state space partition to the new (approxi-

[1]Note that for $|\rho| = |\sigma| = 1$, $\epsilon(\rho, \sigma) = 0$.

mate) probability distribution. An *adaptive state space aggregation* is a method of using different clusterings sequentially in time, where the quality of each clustering is quantified analogously, with the use of (2.7). One has to explicitly define when a partition is no longer to be considered inappropriate and a concrete realisation of this check will be discussed in the next chapter. Finally, since usually a system starts deterministically in a concrete state ($p_0(s) = 1$ for some s), we want to run the model for some time without aggregating it (i.e. using partitions consisting of trivial clusters). The overall procedure is presented in Algorithm 1.

Algorithm 1: Adaptive state space aggregation of DTMC

Input : DTMC (S, P), initial distribution p_0, time horizon k, parameter $noAgg < k$
Output: $\tilde{\mathbf{p}}_\mathbf{k}$, upper bound on $\|\mathbf{e_k}\|_1$

$i = 0$;
while $i \leq noAgg$ **do**
    $\mathbf{p_{i+1}} = \mathbf{p_i} \cdot \mathbf{P}$;
    $i = i + 1$;
end while
$(\Phi, \pi_\mathbf{i}, \mathbf{\Pi}, errAgg, \epsilon) = \text{aggregate}(S, \mathbf{P}, \mathbf{p_i})$;
$\|\mathbf{e}\|_1 = errAgg$;
while $i \leq k$ **do**
    $\|\mathbf{e}\|_1 = \|\mathbf{e}\|_1 + \pi_\mathbf{i} \cdot \epsilon^T$;
    $\pi_\mathbf{i+1} = \pi_\mathbf{i} \cdot \mathbf{\Pi}$;
    $i = i + 1$;
    if *checkPartition*$(\pi_\mathbf{i}) = false$ **then**
        $\tilde{\mathbf{p}}_\mathbf{i} = \text{deaggregate}(\mathbf{\Phi}, \pi_\mathbf{i})$;
        $(\Phi, \pi_\mathbf{i}, \mathbf{\Pi}, errAgg, \epsilon) = \text{aggregate}(S, \mathbf{P}, \tilde{\mathbf{p}}_\mathbf{i})$;
        $\|\mathbf{e}\|_1 = \|\mathbf{e}\|_1 + errAgg$;
    end if
end while
$\tilde{\mathbf{p}}_\mathbf{k} = \text{deaggregate}(\mathbf{\Phi}, \pi_\mathbf{k})$;
return $\tilde{\mathbf{p}}_\mathbf{k}, \|\mathbf{e}\|_1$;

Keep in mind that since a problem of model checking time-bounded specifications of a DTMC is reduced to the problem of its transient analysis, adaptive aggregation can be utilised while working with the corresponding PCTL driven transformation. On a final note, observe that we do not require a normalisation condition $\forall \rho \in \Phi \ \sum_{\sigma \in \Phi} \Pi(\rho, \sigma) = 1$ to be true, i.e. matrix $\mathbf{\Pi}$ might not be 'transition' in a strict mathematical sense. In this case, elements of vectors $\pi_\mathbf{k}$ might not sum to one and therefore such vectors cannot be called 'probability vectors'. Therefore, given Π, abstraction (Φ, Π) might or might not be viewed as a DTMC according to the Definition 1. However, we still want to associate elements of $\mathbf{\Pi}$ with transition probabilities and elements of $\pi_\mathbf{k}$ with transient probabilities in the abstract setting. To avoid any confusion, we will reserve the term 'stochastic' for matrices and vectors that satisfy the corresponding normalisation property. Violation of stochasticity should not discourage us from using such abstractions: functions Π (respectively, π_k) simply serve as higher-level representatives of functions P (respectively, p_k) on a new state space and provide us with approximations of their concrete counterparts.

Example 2. Consider DTMC from Example 1 and let us compute approximation $\tilde{\mathbf{p}}_4$. Denote $\sigma_0 \coloneqq \{s_0\}$, $\sigma_1 \coloneqq \{s_1\}$, $\sigma_{23} \coloneqq \{s_2, s_3\}$ and let the state space partition be $\Phi = \{\sigma_0, \sigma_1, \sigma_{23}\}$. Then $\pi_{\mathbf{0}} = [1, 0, 0]$, i.e. $\tilde{\mathbf{p}}_{\mathbf{0}} = [1, 0, 0, 0]$ and therefore $\|\mathbf{e_0}\|_1 = 0$ – none of the states has effectively changed its transient probability due to aggregation. The corresponding abstract transition matrix is

$$\mathbf{\Pi} = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.4 & 0.2 & 0.2 \\ 1 & 0 & 0.5 \end{bmatrix}.$$

Error factors associated with this partition are:

$$\begin{aligned} \epsilon(\sigma_0) &= 0; \\ \epsilon(\sigma_1) &= \epsilon(\sigma_1, \sigma_{23}) = 0.6; \\ \epsilon(\sigma_{23}) &= \epsilon(\sigma_{23}, \sigma_0) + \epsilon(\sigma_{23}, \sigma_{23}) = 0.5 + 0.1 = 0.6. \end{aligned}$$

Cluster σ_0 produces no error since it and all of its successors are trivial. Error in σ_1 originates from probability forwarding into cluster σ_{23}. Finally, error in cluster σ_{23} comes from 1) probability forwarding from s_2 to previously inaccessible s_0 and 2) propagating probability to itself much differently as compared to unaggregated case. Let $\epsilon \coloneqq [\epsilon(\sigma_0), \epsilon(\sigma_1), \epsilon(\sigma_{23})]$. Now we can perform iterations using vectors and matrices of size 3:

$$\begin{aligned} \|\mathbf{e_1}\|_1 &= \|\mathbf{e_0}\|_1 + \pi_{\mathbf{0}} \cdot \epsilon^T = 0; & \pi_{\mathbf{1}} &= \pi_{\mathbf{0}} \cdot \mathbf{\Pi} = [0.8, 0.2, 0]; \\ \|\mathbf{e_2}\|_1 &= \|\mathbf{e_1}\|_1 + \pi_{\mathbf{1}} \cdot \epsilon^T = 0.12; & \pi_{\mathbf{2}} &= \pi_{\mathbf{1}} \cdot \mathbf{\Pi} = [0.72, 0.2, 0.04]; \\ \|\mathbf{e_3}\|_1 &= \|\mathbf{e_2}\|_1 + \pi_{\mathbf{2}} \cdot \epsilon^T = 0.264; & \pi_{\mathbf{3}} &= \pi_{\mathbf{2}} \cdot \mathbf{\Pi} = [0.696, 0.184, 0.06]; \\ \|\mathbf{e_4}\|_1 &= \|\mathbf{e_3}\|_1 + \pi_{\mathbf{3}} \cdot \epsilon^T = 0.4104; & \pi_{\mathbf{4}} &= \pi_{\mathbf{3}} \cdot \mathbf{\Pi} = [0.6904, 0.176, 0.0668]; \end{aligned}$$

from where $\tilde{\mathbf{p}}_{\mathbf{4}} = [0.6904, 0.176, 0.0334, 0.0334]$. For any state s the point-wise uncertainty is bound by $\|\mathbf{e_4}\|_1$, that is, $|p(s) - \tilde{p}(s)| \leq \|\mathbf{e_4}\|_1$; exact calculations from Example 1 confirm this result. Note that we managed to obtain meaningful approximation, despite the fact that neither of $\mathbf{\Pi}$, $\pi_{\mathbf{4}}$ or $\tilde{\mathbf{p}}_{\mathbf{4}}$ are stochastic.

2.1.3 Threshold Abstraction

Threshold abstraction [7] is yet another approximation technique that can be used for any class of Markov chains and its main idea for DTMCs can be described as follows. Let δ be *the truncation threshold*. We start with a probability distribution p_0 and replace it with the distribution $\hat{p}_0$ by dropping the states that have negligible probability:

$$\hat{p}_0(s) = \begin{cases} p_0(s), & \text{if } p_0(s) \geq \delta \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

We then propagate probability mass we are left with, obtaining $\tilde{p}_1$, and then repeat truncations before successive iterations. The resulting distribution $\tilde{p}_k$ is an underapproximation of the true distribution since the probability mass that was truncated could have remained in a given state or might have been transported to other ones. The total probability loss

$1 - \sum_{s \in S} \tilde{p}_k(s)$ then serves as an (exact) upper bound on the approximation error. Note that this technique can even be used for chains with an infinite state space because during each iteration we deal only with the sets of active/discovered states.

Example 3. Consider again DTMC from Example 1 with initial distribution $\mathbf{p_0} = [1, 0, 0, 0]$. Let $\delta = 0.1$ be the truncation threshold. Then

$$\begin{aligned}
&\hat{\mathbf{p}}_\mathbf{0} = [1, 0, 0, 0]; && \tilde{\mathbf{p}}_\mathbf{1} = \hat{\mathbf{p}}_\mathbf{0} \cdot \mathbf{P} = [0.8, 0.2, 0, 0];\\
&\hat{\mathbf{p}}_\mathbf{1} = [0.8, 0.2, 0, 0]; && \tilde{\mathbf{p}}_\mathbf{2} = \hat{\mathbf{p}}_\mathbf{1} \cdot \mathbf{P} = [0.72, 0.2, 0.08, 0];\\
&\hat{\mathbf{p}}_\mathbf{2} = [0.72, 0.2, 0, 0]; && \tilde{\mathbf{p}}_\mathbf{3} = \hat{\mathbf{p}}_\mathbf{2} \cdot \mathbf{P} = [0.656, 0.184, 0.08, 0];\\
&\hat{\mathbf{p}}_\mathbf{3} = [0.656, 0.184, 0, 0]; && \tilde{\mathbf{p}}_\mathbf{4} = \hat{\mathbf{p}}_\mathbf{3} \cdot \mathbf{P} = [0.5984, 0.168, 0.0736, 0];
\end{aligned}$$

and $\|\mathbf{e_4}\|_1 = 1 - \|\tilde{\mathbf{p}}_\mathbf{4}\|_1 = 0.16$. State space reduction was achieved since during each iteration we have been working with at most three states at a time.

The example above also illustrates one important property of threshold abstraction worth mentioning. Notice how s_1 is constantly sending small portions of probability mass to s_2 that are being immediately truncated. If s_2 was absorbing (for instance, after a PCTL driven transformation during model checking), in the long run, it could have accumulated a significant probability mass, yet constantly truncating these small accruals would lead to an enormous error. In general, this does not happen with state space aggregation: working with clusters allows us to guess where the probability is approximately located. On the other hand, ϵ-terms give us a rather conservative error bound compared to probability loss.

2.2 Continuous-Time Markov Chains

Now we will generalise ideas from the previous section and introduce Markov chains that act in continuous time.

Definition 3. [17] *A continuous-time Markov chain (CTMC)* is a pair $C = (S, R)$, where

- S is *the set of states* and
- $R : S \times S \to \mathbb{R}_{\geq 0}$ is *the transition rate function.*

Similarly as with DTMCs, we will implicitly assume that the set S is finite. R is a function that for each pair of different states assigns a rate used as a parameter of an exponential distribution. Formally, a transition from state r to state s can occur if $R(r, s) > 0$ and the probability that this transition is triggered within t time units is $1 - e^{-R(r,s) \cdot t}$. A probabilistic choice arises through race condition when for current state r there exist several states s such that $R(r, s) > 0$, the first transition triggered then defines the next state of a CTMC. The time spent in state r, before any such transition occurs, is exponentially distributed with parameter $E(r) \coloneqq \sum_{s \in S} R(r, s)$, which is called *the exit rate* of state r. States for which $E(r) = 0$ are called *absorbing*. A CTMC is initialised via the distribution $p_0(s) \coloneqq \mathbb{P}(C(0) = s)$. Similar to DTMCs, it is sometimes helpful to visualise CTMCs as a graph structure where edges represent non-zero transition rates, as Figure 2.4 suggests.

Definition 4. Let $C = (S, R)$ be CTMC. *An infinitesimal generator function* $Q : S \times S \to \mathbb{R}$ of C is defined as

$$Q(r, s) = \begin{cases} -E(r), & \text{if } r = s \\ R(r, s), & \text{otherwise.} \end{cases}$$

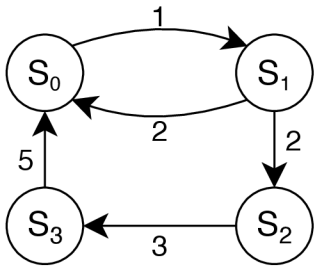

Figure 2.4: A simple CTMC.

Definition 5. Let $C = (S, R)$ be CTMC and Q be its infinitesimal generator function. Let $q \geq \max_{s \in S} E(s)$ be *uniformisation rate*. A *uniformised DTMC of* C *given uniformisation rate* q is a DTMC (S, unif_R^q) having transition probability function unif_R^q defined as

$$\mathrm{unif}_R^q(r,s) = \begin{cases} 1 + \frac{Q(r,s)}{q}, & \text{if } r = s \\ \frac{Q(r,s)}{q}, & \text{otherwise.} \end{cases}$$

A uniformised DTMC (S, unif_R^q) serves as a time-discretisation of the CTMC $C = (S, R)$ with respect to the fastest event with rate q that can occur. Usually, we pick q to be the maximum exit rate in S, which corresponds to the shortest mean residence time in the system, although any value larger or equal tso this rate will be sufficient. If r is the current state and $r \neq s$, then $\mathrm{unif}_R^q(r,s) \coloneqq \frac{Q(r,s)}{q} = \frac{R(r,s)}{q}$ yields the probability of triggering the transition from r to s given that *any* discrete transition (i.e. including the one from r to r) occurs. Therefore, the complement $1 - \sum_{s \in S, r \neq s} \frac{R(r,s)}{q} = 1 - \frac{E(r)}{q} = 1 + \frac{Q(r)}{q} =: \mathrm{unif}_R^q(r,r)$ gives the self-loop transition probability from r to r. Similarly as in the previous section, we will use symbols $\mathbf{R}$, $\mathbf{Q}$ or $\mathbf{unif_R^q}$ to denote matrices associated with the corresponding functions. We are interested in computing the transient probability distribution $p_t(s) \coloneqq \mathbb{P}(C(t) = s)$ for any $t > 0$, which is the interest of the uniformisation procedure.

2.2.1 Uniformisation

Any Markov chain can be viewed [7] as a stochastic process $X = \{X(t), t \in T\}$ defined on a (discrete) state space S, acting in time domain T and satisfying the Markov property. Probability $\mathbb{P}(X(t) = s), s \in S, t \in T$ denotes the probability of residing in state s at time t. The choice of a domain T determines the type of the chain we are dealing with: $\mathbb{N}_0$ for DTMCs, $\mathbb{R}_{\geq 0}$ for CTMCs.

The main idea of a uniformisation method is to split CTMC $C = \{C(t), t \in \mathbb{R}_{\geq 0}\} = (S, R)$ into two independent stochastic processes: $\{D_C(k), k \in \mathbb{N}_0\}, D_C(k) \in S$ and $\{B_C(t), t \in \mathbb{R}_{\geq 0}\}$, $B_C(t) \in \mathbb{N}_0$ such that

$$p_t(s) \coloneqq \mathbb{P}(C(t) = s) = \mathbb{P}(D_C(B_C(t)) = s).$$

Notice that D_C is a DTMC defined on the same state space as C and B_C is a CTMC with an infinite state space. We will refer to B_C as a birth process associated with C. In the case where S is finite and all $R(\cdot,\cdot)$ are bounded, the existence of such processes is guaranteed. From the independence of D_C and B_C, we obtain:

$$\mathbb{P}(D_C(B_C(t)) = s) = \sum_{k=0}^{\infty} \mathbb{P}(D_C(k) = s) \cdot \mathbb{P}(B_C(t) = k). \tag{2.9}$$

Intuition behind this expression is as follows. Expression $\mathbb{P}(D_C(k) = s)$ represents a probability that C resides at state s after k 'discrete' jumps. Since we cannot know in advance how many steps will be performed, we invoke the total probability theorem [3], where $\mathbb{P}(B_C(t) = k)$ captures a probability of performing k such steps within t continuous time units. Intuitively, D_C keeps track of the current state of C and B_C keeps track of an elapsed time (in probabilistic sense). Let us introduce shortcuts $u_k(s) \coloneqq \mathbb{P}(D_C(k) = s)$ and $\beta_k \coloneqq \mathbb{P}(B_C(t) = k)$ A concrete choice of D_C and B_C is of interest of the concrete uniformisation procedure.

Definition 6. Let $C = (S, R)$ be CTMC. *Standard uniformisation (SU)* is a splitting (D_C, B_C) according to the following rules:

- $q \geq \max_{s \in S} E(s)$ is a uniformisation rate.
- $B_C = (\mathbb{N}_0, R_{B_C})$ is a CTMC s.t. $\mathbb{P}(B_C(0) = 0) = 1$ and R_{B_C} is defined as:

$$R_{B_C}(i, j) = \begin{cases} q, & \text{if } j = i + 1 \\ 0, & \text{otherwise.} \end{cases}$$

- $D_C = (S, \mathrm{unif}_R^q)$ is a uniformised DTMC given rate q that has the same initial probability distribution as C.

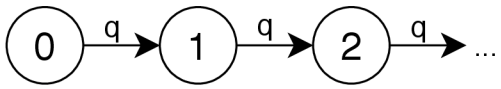

Figure 2.5: Poisson process with rate q.

We recognise B_C to be *a pure birth process with constant rate* q (i.e. Poisson with rate q), see Figure 2.5, for which analytical solution is known to be:

$$\beta_k = e^{-qt} \frac{(qt)^k}{k!} =: \psi_{qt}(k), k \in \mathbb{N}_0.$$

Therefore, (2.9) becomes

$$p_t(s) = \sum_{k=0}^{\infty} u_k(s) \cdot \psi_{qt}(k). \tag{2.10}$$

Finally, for a given precision ε_{fg}, an iterative scheme of Fox and Glynn [10] can provide bounds L, R such that

$$1 - \varepsilon_{fg} \leq \sum_{k=L}^{R} \psi_{qt}(k). \tag{2.11}$$

We can then truncate the infinite sum in (2.10) to obtain an underapproximation of the true probability distribution:

$$\hat{p}_t(s) \coloneqq \sum_{k=L}^{R} u_k(s) \cdot \psi_{qt}(k) \leq p_t(s) \tag{2.12}$$

Combining (2.11) and (2.12), we arrive at the conclusion

$$\begin{aligned} \|\hat{\mathbf{p}}_\mathbf{t}\|_1 &= \sum_{s \in S} \hat{p}_t(s) = \sum_{s \in S} \sum_{k=L}^{R} u_k(s) \psi_{qt}(k) = \sum_{k=L}^{R} \sum_{s \in S} u_k(s) \psi_{qt}(k) \\ &= \sum_{k=L}^{R} \psi_{qt}(k) \sum_{s \in S} u_k(s) = \sum_{k=L}^{R} \psi_{qt}(k) \\ &\geq 1 - \varepsilon_{fg}, \end{aligned}$$

and therefore

$$\|\mathbf{e}_\mathbf{t}\|_1 = \|\mathbf{p}_\mathbf{t} - \hat{\mathbf{p}}_\mathbf{t}\|_1 = \|\mathbf{p}_\mathbf{t}\|_1 - \|\hat{\mathbf{p}}_\mathbf{t}\|_1 = 1 - \|\hat{\mathbf{p}}_\mathbf{t}\|_1 \leq \varepsilon_{fg},$$

that is, we have a guarantee that the total probability loss that comes from truncation will not exceed ε_{fg}. Notice that this proposition holds since transient probabilities $u_k(s)$ sum to one.

Example 4. Consider CTMC from Figure 2.4 having initial distribution $\mathbf{p_0} = [1, 0, 0, 0]$. Its transition rate matrix is

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 5 & 0 & 0 & 0 \end{bmatrix}.$$

We are interested in finding the transient probability distribution for this chain at time $t = 0.4$. The maximum exit rate is the one of s_3, so $q = 5$[2]. Let $\varepsilon_{fg} = 0.1$. For this accuracy and the product $q \cdot t = 2$, Fox-Glynn procedure would return $L = 0, R = 4$ and $\psi_{\mathbf{q \cdot t}} \approx [0.1353, 0.2706, 0.2706, 0.1804, 0.0902]$, where displayed values are truncated to four decimal places. R being equal to 4 means that we will need to find distributions $\mathbf{u_0}, \ldots, \mathbf{u_4}$ for the uniformised DTMC. L being equal to zero means that we start to weigh these distributions starting from the initial one. Finally, the vector $\psi_{\mathbf{q \cdot t}}$ contains all five Poisson probabilities needed to perform this weighing from 0 to 4. Let $\mathbf{s_i} \coloneqq \sum_{k=0}^{i} \mathbf{u_k} \cdot \psi_{qt}(k)$ denote the partial sum. The uniformisation of $\mathbf{R}$ with rate $q = 5$ is the discrete chain from

[2] Any number larger or equal to $\max_{s \in S}\{E(s)\}$ will do. In practice, we often choose maximum exit rate multiplied by 1.02, for numerical reasons.

Example 1 where we have already computed its probability distributions. Therefore:

$$\begin{aligned}
\mathbf{s_0} &= \mathbf{u_0} \cdot \psi_{qt}(0) \approx [0.1353, 0, 0, 0];\\
\mathbf{s_1} &= \mathbf{s_0} + \mathbf{u_1} \cdot \psi_{qt}(1) \approx [0.3518, 0.0541, 0, 0];\\
\mathbf{s_2} &= \mathbf{s_1} + \mathbf{u_2} \cdot \psi_{qt}(2) \approx [0.5467, 0.1082, 0.0216, 0];\\
\mathbf{s_3} &= \mathbf{s_2} + \mathbf{u_3} \cdot \psi_{qt}(3) \approx [0.6651, 0.1414, 0.0418, 0.0086];\\
\mathbf{s_4} &= \mathbf{s_3} + \mathbf{u_4} \cdot \psi_{qt}(4) \approx [0.7234, 0.1566, 0.0525, 0.0147].
\end{aligned}$$

Finally, $\hat{\mathbf{p}}_\mathbf{t} = \mathbf{s_4}$ and $\|\mathbf{e_t}\|_1 = 1 - \|\hat{\mathbf{p}}_\mathbf{t}\|_1 = 0.0526$; we confirm that $\|\mathbf{e_t}\|_1 \leq \varepsilon_{fg}$.

The main drawback of SU is that for large uniformisation rates q the mean of the Poisson distribution $\psi_{qt}(\cdot)$ is large and so is the upper truncation point R. This means that to find the solution of $C(t)$ one must perform plenty of iterations for the process $D_C(k)$. Adaptive uniformisation solves this issue by allowing the rates of the birth process to change in each step.

Definition 7. Let $C = (S, R)$ be CTMC. *Adaptive uniformisation (AU)* [7] is a splitting (D_C, B_C) accoring to the following rules:

- Let $q_0, q_1, ...$ be an infinite sequence of uniformisation rates satisfying

$$q_i \geq \max\{E(s) \mid s \in S, u_i(s) > 0\}. \tag{2.13}$$

- $B_C = (\mathbb{N}_0, R_{B_C})$ is a CTMC s.t. $\mathbb{P}(B_C(0) = 0) = 1$ and R_{B_C} is defined as:

$$R_{B_C}(i,j) = \begin{cases} q_i, & \text{if } j = i+1 \\ 0, & \text{otherwise.} \end{cases}$$

- $D_C = (S, \text{unif}_R^{q_i})$ is a DTMC whose transition probability matrix during time step i is a uniformisation of R with rate q_i.

We start at discrete time 0 with a subset of states in S that have non-zero initial probability $u_0(\cdot)$, such states will be called *active* or *significant*. The largest exit rate q_0 from the states within this subset is to be the (local) uniformisation rate. We then compute $\text{unif}_R^{q_0}$ to be the transition probability matrix for process D_C at time 0, perform probability propagation and obtain $u_1(\cdot)$. We then repeat the procedure of defining the subset of active states, finding (local) uniformisation rate q_1, uniformising the rate matrix according to this rate and propagating probability. This way we obtain a sequence $q_0, q_1, ...$ and can construct a birth process, see Figure 2.6. In order to solve this CTMC, we apply SU. Notice that $\forall i \in \mathbb{N}_0\ q_i < q$ where $q \geq max_{s \in S} E(s)$ – we can use q as a (global) uniformisation rate for B_C and its solution is

$$\beta_k = \sum_{l=0}^{\infty} \mathbb{P}(D_{B_C}(l) = k) \cdot \psi_{qt}(l), \tag{2.14}$$

where D_{B_C} is a uniformised DTMC associated with B_C, see Figure 2.6. Observe that for $k = 0$:

$$\mathbb{P}(D_{B_C}(l) = 0) = \mathbb{P}(D_{B_C}(l-1) = 0) \cdot \left(1 - \frac{q_0}{q}\right). \tag{2.15}$$

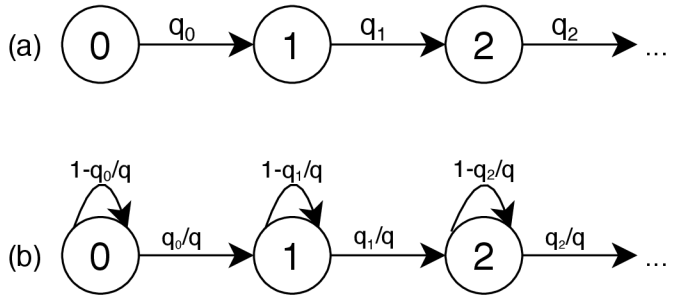

Figure 2.6: A general birth process (a) and its uniformised DTMC (b).

and for $k > 0$:

$$\mathbb{P}(D_{B_C}(l) = k) = \mathbb{P}(D_{B_C}(l-1) = k-1) \cdot \frac{q_{k-1}}{q} + \mathbb{P}(D_{B_C}(l-1) = k) \cdot \left(1 - \frac{q_k}{q}\right). \quad (2.16)$$

Hence, to compute β_k, only rates $q_0, ..., q_k$ must be known: the computation of $u_k(\cdot)$ and β_k can be interleaved. Combining (2.9) and (2.14), we are able to compute transient probabilities for CTMC C:

$$p_t(s) = \sum_{k=0}^{\infty} u_k(s) \cdot \beta_k = \sum_{k=0}^{\infty} u_k(s) \cdot \sum_{l=0}^{\infty} \mathbb{P}(D_{B_C}(l) = k) \cdot \psi_{qt}(l). \quad (2.17)$$

The inner infinite sum can be truncated using the Fox-Glynn scheme for a given precision ε_{fg}. The outer summing can be stopped after step R' when $\sum_{k=0}^{R'} \beta_k \geq 1 - \varepsilon_{bp}$ for a given precision $\varepsilon_{bp} < \varepsilon_{fg}$, and so (2.17) becomes

$$\hat{p}_t(s) \coloneqq \sum_{k=0}^{R'} u_k(s) \cdot \sum_{l=L}^{R} \mathbb{P}(D_{B_C}(l) = k) \cdot \psi_{qt}(l). \quad (2.18)$$

Both truncations lead to an underapproximation of the true probability distribution and the total error is given by the probability loss $1 - \|\hat{\mathbf{p}}_\mathbf{t}\|_1$ with an a priori specified bound ε_{bp}.

Example 5. Consider again CTMC (S, R) and time bound t from Example 4 and let $\varepsilon_{fg} = \varepsilon_{bp} = 0.1$; global uniformisation rate q, truncation bounds L, R and a Poisson distribution $\psi_{qt}(\cdot)$ for the inner birth process remain the same. Let $\mathbf{s_i} \coloneqq \sum_{k=0}^{i} \mathbf{u_k} \cdot \beta_k$ denote the partial sum. Let $\mathbf{v_k} \coloneqq [\mathbb{P}(D_{B_C}(0) = k), ..., \mathbb{P}(D_{B_C}(R) = k)]$ be the solution of the birth process for the 'outer' discrete time horizon k; note that $\mathbf{v_k}$ is *not* a probability distribution and its entries are not supposed to sum to one.
Iteration $i = 0$:

1. initial distribution for uniformised DTMC is $\mathbf{u_0} = [1, 0, 0, 0]$;
2. s_0 is the only active state, therefore, $q_0 = E(s_0) = 1$;
3. using (2.15), we obtain $\mathbf{v_0} = [1, 0.8, 0.64, 0.512, 0.4096]$;

4. $\beta_0 = \mathbf{v_0} \cdot {\psi_{\mathbf{qt}}}^T = 0.6544$;
5. $\mathbf{s_0} = \mathbf{u_0} \cdot \beta_0 = [0.6544, 0, 0, 0]$;
6. $\sum_{k=0}^{0} \beta_k = 0.6544 < \varepsilon_{bp}$, so we need to push probability and move on to the next iteration;
7. $\mathbf{u_1} = \mathbf{u_0} \cdot \mathbf{unif}_{\mathbf{R}}^{\mathbf{q_0}} = [0, 1, 0, 0]$.

Iteration $i = 1$:

1. s_1 is the only active state, so $q_1 = E(s_1) = 4$;
2. using (2.16), we obtain $\mathbf{v_1} = [0, 0.2, 0.2, 0.168, 0.136]$;
3. $\beta_1 = \mathbf{v_1} \cdot {\psi_{\mathbf{qt}}}^T = 0.1508$;
4. $\mathbf{s_1} = \mathbf{s_0} + \mathbf{u_1} \cdot \beta_1 = [0.6544, 0.1508, 0, 0]$;
5. $\sum_{k=0}^{1} \beta_k = 0.8052 < \varepsilon_{bp}$, more iterations needed;
6. $\mathbf{u_2} = \mathbf{u_1} \cdot \mathbf{unif}_{\mathbf{R}}^{\mathbf{q_1}} = [0.5, 0, 0.5, 0]$.

Iteration $i = 2$:

1. s_0 and s_2 are active, so $q_2 = \max\{E(s_0), E(s_2)\} = \max\{1, 3\} = 3$;
2. using (2.16), we obtain $\mathbf{v_2} = [0, 0, 0.16, 0.224, 0.224]$;
3. $\beta_2 = \mathbf{v_2} \cdot {\psi_{\mathbf{qt}}}^T = 0.1039$;
4. $\mathbf{s_2} = \mathbf{s_1} + \mathbf{u_2} \cdot \beta_2 = [0.7064, 0.1508, 0.0519, 0]$;
5. $\sum_{k=0}^{2} \beta_k = 0.9092 > \varepsilon_{bp}$, we can now stop iterations.

Finally, $\hat{\mathbf{p}}_\mathbf{t} = \mathbf{s_2}$ and $\|\mathbf{e_t}\|_1 = 1 - \|\hat{\mathbf{p}}_\mathbf{t}\|_1 = 0.0907$. We confirm that $\|\mathbf{e_t}\|_1 \leq \varepsilon_{bp}$. Notice that, instead of four vector-matrix multiplications performed in 4, we needed only two.

The main advantage of AU lies in the fact that uniformisation rates q_i are 'discovered' with probability propagation and there is a chance that at a given time i, $q_i \ll q$, which allows a birth process to jump at lower rates and therefore it is possible that R' will be substantially lower than R. Numerically, this allows to perform much less vector-matrix multiplications to solve for D_C and at the same time arrive at the same result with the same accuracy as SU[3]. Furthermore, threshold abstraction can be used with both SU or AU to solve for D_C, although it is particularly favourable with AU since truncating the state space using threshold $\delta > 0$ leads to smaller subsets of active states and a birth process can jump at even slower rates q_i. In the sequel, we will refer to the combination of SU with threshold abstraction as *fast SU (FSU)*, and to the combination of AU with threshold abstraction as *fast AU (FAU)*[4]. Also note that in this case probability loss is the only way to estimate the approximation error since an a priori specified error bound cannot be guaranteed.

[3]One could argue that the complexity of (2.12), which is R, is substantially lower than that of (2.18), which is $R' \cdot R$, since during each step we must solve for a birth process. However, due to its extremely simple structure, this computation is trivial compared to a vector-matrix multiplication needed to compute $u_k(\cdot)$. However, as will be shown later, in some cases solving for B_C can become more noticeable. The bottom line here is that AU will always demonstrate better performance than SU, although this increase may not correlate with the reduction ratio R/R', see experiments at the end of Section 4.2.

[4]In this case FSU can be used while solving a birth process to further decrease complexity of the computation.

2.2.2 Model Checking and Aggregating CTMCs

All the model checking procedures defined for DTMCs in Subsection 2.1.1 automatically translate into the continuous case and can be applied in combination with uniformisation techniques. In the case of CTMCs, the specification language is *Continuous Stochastic Logic (CSL)* that is also based on CTL. For the same reasons mentioned earlier, we will restrict ourselves to its subset and consider only operators 'eventually', $\Diamond^{\leq t} A$, and 'always', $\Box^{\leq t} A$. Their semantics is defined analogously as in the discrete case, except that bound t (again, strictly finite) is now considered on a continuous time-domain. A *CSL driven transformation* is, again, constructed by forcing states in $Sat(A)$ to be absorbing. The desired probability $\mathbb{P}(\Diamond^{\leq t} A)$ (respectively, $\mathbb{P}(\Box^{\leq t} A)$) for a given CTMC can be found using uniformisation to compute transient probabilities at time t of its CSL driven transformation and summing the resulting state probabilities over the set $Sat(A)$. The resulting value is an underapproximation of the true probability and the upper error bound is again given by $1 - \|\hat{\mathbf{p}}_{\mathbf{t}}^{\mathbf{C[A]}}(\mathbf{s})\|_1$, since this value encompasses a total probability loss for each of the states in S. A comprehensive description of CSL model checking can be found in [17].

As to the state space aggregation, one could suspect that it can be integrated with e.g. standard uniformisation while solving for uniformised DTMC. This idea was outlined in [1] and will be reviewed in Chapter 4, where we will formalise this approach and rigorously derive bounds on the approximation error.

Chapter 3

Adaptive Aggregation for DTMCs

The goal of this chapter is to develop ideas presented in [1] and introduce a notion of a general state-space aggregation for discrete-time Markov models. This will allow us to explore various aggregation schemes as well as obtain a better approximation error estimate.

Definition 8. Let $D = (S, P)$ be DTMC with initial distribution p_0. Let $\tilde{P} : S \times S \to \mathbb{R}$ be any real function relating each pair of states, let $\tilde{p}_0 : S \to \mathbb{R}$ be a function that to each state assigns any real number and let functions $\tilde{p}_k : S \to \mathbb{R}, k > 0$ be recursively defined as

$$\tilde{p}_k(s) = \sum_{r \in S} \tilde{p}_{k-1}(r)\tilde{P}(r, s) \tag{3.1}$$

or, using matrix notation:

$$\tilde{\mathbf{p}}_\mathbf{k} = \tilde{\mathbf{p}}_{\mathbf{k-1}} \cdot \tilde{\mathbf{P}}. \tag{3.2}$$

Function $\tilde{P}$ will be referred to as *an approximation of* P, functions $\tilde{p}_k$ as *approximations of* p_k and a structure $(S, \tilde{P})$ will be referred to as *an approximation of DTMC* D.

We are also interested in computing an approximation error $e_k(s) \coloneqq \tilde{p}_k(s) - p_k(s)$. First, note that $e_0(s) = \tilde{p}_0(s) - p_0(s)$ captures an error associated with using $\tilde{p}_0$ instead of p_0. For $k > 0$, we argue as follows:

$$\begin{aligned}
\tilde{p}_k(s) =& \sum_{r \in S} \tilde{p}_{k-1}(r)\tilde{P}(r, s) \\
\Leftrightarrow& \\
p_k(s) + e_k(s) =& \sum_{r \in S} (p_{k-1}(r) + e_{k-1}(r)) \left(\tilde{P}(r, s) - P(r, s) + P(r, s)\right) \\
=& \sum_{r \in S} p_{k-1}(r)P(r, s) + \sum_{r \in S} e_{k-1}(r)P(r, s) + \\
&+ \sum_{r \in S} (p_{k-1}(r) + e_{k-1}(r)) \left(\tilde{P}(r, s) - P(r, s)\right) \\
=& p_k(s) + \sum_{r \in S} e_{k-1}(r)P(r, s) + \sum_{r \in S} \tilde{p}_{k-1}(r) \left(\tilde{P}(r, s) - P(r, s)\right),
\end{aligned}$$

from where

$$e_k(s) = \sum_{r \in S} e_{k-1}(r) P(r,s) + \sum_{r \in S} \tilde{p}_{k-1}(r) \left(\tilde{P}(r,s) - P(r,s) \right). \tag{3.3}$$

This recursive formula gives us an insight into how an error is generated when we approximate a DTMC. The first summand represents a casual propagation of existing error as if we were using exact transitions $P(r,s)$ for probability propagation. On the other hand, the term $\tilde{p}_{k-1}(r) \left(\tilde{P}(r,s) - P(r,s) \right)$ captures an error that is generated in each step between states r and s while using an approximation $\tilde{P}(r,s)$ instead of $P(r,s)$; the sum over all states then yields the total error generated into state s. If $\mathbf{e_k}$ is a row vector associated with function e_k, then for its L_1-norm we obtain:

$$\begin{aligned}
\|\mathbf{e_k}\|_1 &= \sum_{s \in S} |e_k(s)| \\
&= \sum_{s \in S} \left| \sum_{r \in S} e_{k-1}(r) P(r,s) + \sum_{r \in S} \tilde{p}_{k-1}(r) \left(\tilde{P}(r,s) - P(r,s) \right) \right| \\
&\leq \sum_{s \in S} \left| \sum_{r \in S} e_{k-1}(r) P(r,s) \right| + \sum_{s \in S} \left| \sum_{r \in S} \tilde{p}_{k-1}(r) \left(\tilde{P}(r,s) - P(r,s) \right) \right|.
\end{aligned}$$

Let us inspect the first term:

$$\begin{aligned}
\sum_{s \in S} \left| \sum_{r \in S} e_{k-1}(r) P(r,s) \right| &\leq \sum_{s \in S} \sum_{r \in S} |e_{k-1}(r)| \, P(r,s) = \sum_{r \in S} \sum_{s \in S} |e_{k-1}(r)| \, P(r,s) \\
&= \sum_{r \in S} |e_{k-1}(r)| \sum_{s \in S} P(r,s) = \sum_{r \in S} |e_{k-1}(r)| = \|\mathbf{e_{k-1}}\|_1,
\end{aligned}$$

and therefore update equation for the L_1-norm of the error vector at time k becomes

$$\|\mathbf{e_k}\|_1 \leq \|\mathbf{e_{k-1}}\|_1 + \sum_{s \in S} \left| \sum_{r \in S} \tilde{p}_{k-1}(r) \left(\tilde{P}(r,s) - P(r,s) \right) \right|, \tag{3.4}$$

where

$$\|\mathbf{e_0}\|_1 = \sum_{s \in S} |\tilde{p}_0(s) - p_0(s)| \, .$$

Let us now introduce a special class of approximate DTMCs.

Definition 9. Let $D = (S, P)$ be DTMC and let Φ be a clustering of S. Let $\Pi : \Phi \times \Phi \to \mathbb{R}$ be any real function relating each pair of clusters and define $\pi_0(\sigma) \coloneqq \sum_{s \in \sigma} p_0(s)$. Then an approximation $(S, \tilde{P})$ of DTMC D where $\tilde{P}(r,s) = \frac{\Pi(\rho,\sigma)}{|\sigma|}, r \in \rho, s \in \sigma$ and $\tilde{p}_0(s) = \frac{\pi_0(\sigma)}{|\sigma|}$, $s \in \sigma$ will be referred to as *a state-space aggregation of* D given *abstract state space* Φ.

Notice that

$$s, s' \in \sigma \Rightarrow \tilde{p}_0(s) = \tilde{p}_0(s') = \frac{\pi_0(\sigma)}{|\sigma|}.$$

Similarly,

$$(r, r' \in \rho), (s, s' \in \sigma) \Rightarrow \tilde{P}(r, s) = \tilde{P}(r', s') = \frac{\Pi(\rho, \sigma)}{|\sigma|},$$

and therefore update equation (3.1) yields

$$s, s' \in \sigma \Rightarrow \tilde{p}_k(s) = \sum_{r \in S} \tilde{p}_{k-1}(r)\tilde{P}(r, s) = \sum_{r \in S} \tilde{p}_{k-1}(r)\tilde{P}(r, s') = \tilde{p}_k(s').$$

Three equalities above illustrate an important property of aggregation: any two states from two given clusters have the same approximate transition probability $\tilde{P}(\cdot, \cdot)$ and any two states in a given cluster at any time step $k \geq 0$ share the same value of approximate transient probability $\tilde{p}_k(\cdot)$. Hence, instead of computing transient probabilities for each of the states, we can compute them for a single state within each cluster or, equivalently, for a cluster as a whole. Let $\pi_k(\sigma) \coloneqq \sum_{s \in \sigma} \tilde{p}_k(s)$, $k > 0$; conversely, $\tilde{p}_k(s) = \frac{\pi_k(\sigma)}{|\sigma|}$, $s \in \sigma$, $k \geq 0$, due to Definition 9 and the argument above. Then for $k > 0$:

$$\begin{aligned}
\pi_k(\sigma) &= \sum_{s \in \sigma} \tilde{p}_k(s) = \sum_{s \in \sigma} \sum_{r \in S} \tilde{p}_{k-1}(r)\tilde{P}(r, s) \\
&= \sum_{s \in \sigma} \sum_{\rho \in \Phi} \sum_{r \in \rho} \tilde{p}_{k-1}(r)\tilde{P}(r, s) = \sum_{s \in \sigma} \sum_{\rho \in \Phi} \sum_{r \in \rho} \frac{\pi_{k-1}(\rho)}{|\rho|} \frac{\Pi(\rho, \sigma)}{|\sigma|} \\
&= \sum_{\rho \in \Phi} \sum_{s \in \sigma} \sum_{r \in \rho} \frac{\pi_{k-1}(\rho)}{|\rho|} \frac{\Pi(\rho, \sigma)}{|\sigma|} = \sum_{\rho \in \Phi} \frac{\pi_{k-1}(\rho)}{|\rho|} \frac{\Pi(\rho, \sigma)}{|\sigma|} \sum_{s \in \sigma} \sum_{r \in \rho} 1 \\
&= \sum_{\rho \in \Phi} \pi_{k-1}(\rho)\Pi(\rho, \sigma),
\end{aligned}$$

or in the matrix notation:

$$\pi_{\mathbf{k}} = \pi_{\mathbf{k-1}} \cdot \mathbf{\Pi}. \tag{3.5}$$

Hence, we can operate with vectors $\pi_{\mathbf{k}}$ instead of $\tilde{\mathbf{p}}_{\mathbf{k}}$. As to an error associated with this aggregation, the second term in (3.4) becomes

$$
\begin{aligned}
&\sum_{s\in S}\left|\sum_{r\in S}\tilde{p}_{k-1}(r)\left(\tilde{P}(r,s)-P(r,s)\right)\right| \\
&\quad=\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\sum_{\rho\in\Phi}\sum_{r\in\rho}\tilde{p}_{k-1}(r)\left(\tilde{P}(r,s)-P(r,s)\right)\right| \\
&\quad=\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\sum_{\rho\in\Phi}\sum_{r\in\rho}\frac{\pi_{k-1}(\rho)}{|\rho|}\left(\frac{\Pi(\rho,\sigma)}{|\sigma|}-P(r,s)\right)\right| \\
&\quad=\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\sum_{\rho\in\Phi}\frac{\pi_{k-1}(\rho)}{|\rho|}\sum_{r\in\rho}\left(\frac{\Pi(\rho,\sigma)}{|\sigma|}-P(r,s)\right)\right| \\
&\quad=\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\sum_{\rho\in\Phi}\frac{\pi_{k-1}(\rho)}{|\rho|}\left(\frac{|\rho|}{|\sigma|}\Pi(\rho,\sigma)-\sum_{r\in\rho}P(r,s)\right)\right| \\
&\quad=\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\sum_{\rho\in\Phi}\pi_{k-1}(\rho)\left(\frac{\Pi(\rho,\sigma)}{|\sigma|}-\frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right)\right| \\
&\quad\leq\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\sum_{\rho\in\Phi}\pi_{k-1}(\rho)\left|\frac{\Pi(\rho,\sigma)}{|\sigma|}-\frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right| \\
&\quad=\sum_{\rho\in\Phi}\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\pi_{k-1}(\rho)\left|\frac{\Pi(\rho,\sigma)}{|\sigma|}-\frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right| \\
&\quad=\sum_{\rho\in\Phi}\pi_{k-1}(\rho)\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|}-\frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right|
\end{aligned}
$$

If we denote $\tau(\rho,\sigma)\coloneqq\sum_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|}-\frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right|$ and $\tau(\rho)\coloneqq\sum_{\sigma\in\Phi}\tau(\rho,\sigma)$, then for the L_1-norm of the error vector we obtain

$$
\|\mathbf{e_k}\|_1\leq\|\mathbf{e_{k-1}}\|_1+\sum_{\rho\in\Phi}\pi_{k-1}(\rho)\tau(\rho). \tag{3.6}
$$

Equations (3.5) and (3.6) allow us to compute approximation of the transient probability distribution as well as an upper bound on the L_1-norm of the error vector. Note that neither $\Pi(\cdot,\cdot)$ nor $\tau(\cdot)$ change as time progresses: these values can be computed only once before the first propagation and in the case where $|\Phi|\ll|S|$, using (3.5) and (3.6) instead of (3.2) and (3.4) will be more efficient. We have arrived at the same framework for DTMC approximate analysis as the one described in Section 2.1.2. What differs, however, is that newly derived procedures apply to *any* aggregation scheme Π, not only to (2.3). This allows us to experiment with different strategies and find the most suitable one. Furthermore, observe that

$$\tau(\rho,\sigma) := \sum_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right| \le |\sigma|\max_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right|$$
$$= \max_{s\in\sigma}\left|\Pi(\rho,\sigma) - \frac{|\sigma|}{|\rho|}\sum_{r\in\rho}P(r,s)\right| =: \epsilon(\rho,\sigma)$$

and therefore (3.6) gives us a better error estimate than existing bound in (2.7). To conclude, we are now able to apply arbitrary aggregation scheme Π and compute approximation error with a better error bound. Let us introduce several aggregation schemas that will be of great interest later on, once we establish the principal aggregation algorithm:

- state-space aggregation based on *incoming* transition probabilities [1]:

$$\Pi_{in}(\rho,\sigma) = \frac{1}{|\sigma|}\sum_{a\in\rho}\sum_{s\in\sigma}P(r,s), \tag{3.7}$$

- state-space aggregation based on *outgoing* transition probabilities:

$$\Pi_{out}(\rho,\sigma) = \frac{1}{|\rho|}\sum_{a\in\rho}\sum_{s\in\sigma}P(r,s), \tag{3.8}$$

- *median-based* state-space aggregation:

$$\Pi_{med}(\rho,\sigma) = \frac{|\sigma|}{|\rho|}\operatorname*{med}_{s\in\sigma}\left\{\sum_{r\in\rho}P(r,s)\right\}. \tag{3.9}$$

It is necessary to explain where each of this schemas comes from. The first one originates from [1] and was in detail described in 2.1.2. Inspired by this approach, the second scheme was designed with one specific goal in mind, namely, observe that for each cluster ρ:

$$\sum_{\sigma\in\Phi}\Pi_{out}(\rho,\sigma) = \sum_{\sigma\in\Phi}\frac{1}{|\rho|}\sum_{r\in\rho}\sum_{s\in\sigma}P(r,s) = \frac{1}{|\rho|}\sum_{r\in\rho}\sum_{\sigma\in\Phi}\sum_{s\in\sigma}P(r,s)$$
$$= \frac{1}{|\rho|}\sum_{r\in\rho}\sum_{s\in S}P(r,s) = \frac{1}{|\rho|}\sum_{r\in\rho}1 = 1,$$

i.e. matrix Π_{out} is stochastic, so vectors $\pi_{\mathbf{k}}$ (and therefore $\tilde{\mathbf{p}}_{\mathbf{k}}$) are stochastic as well and the abstract chain (Φ,Π_{out}) is the DTMC in view of Definition 1. This subtle difference has two large benefits. First, from the technical standpoint, this leads to a slightly better approximation compared to (3.7), as will be shown later. Second, preserving stochasticity of $\tilde{\mathbf{p}}_{\mathbf{k}}$ will be the key to safely use uniformisation method and Fox-Glynn algorithm when dealing with CTMCs.

The median-based scheme was derived using the following argument. Assume a specific state space clustering is given. We can arbitrarily define our abstract transition probabilities $\Pi(\cdot,\cdot)$ and the approximation error accrual in each iteration will be captured by (3.6). In order to minimize this accrual, it is sufficient to minimize each of the

$$\tau(\rho,\sigma) \coloneqq \sum_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho}P(r,s)\right|$$

by picking a suitable $\Pi(\rho,\sigma)$. We can safely pull $|\sigma|$ in denominators out of the absolute value and instead minimize

$$\sum_{s\in\sigma}\left|\Pi(\rho,\sigma) - \frac{|\sigma|}{|\rho|}\sum_{r\in\rho}P(r,s)\right|.$$

We recognize this as a problem of minimizing the sum of the absolute deviations, for which solution is known [22] to be

$$\Pi(\rho,\sigma) = \operatorname*{med}_{s\in\sigma}\left\{\frac{|\sigma|}{|\rho|}\sum_{r\in\rho}P(r,s)\right\} = \frac{|\sigma|}{|\rho|}\operatorname*{med}_{s\in\sigma}\left\{\sum_{r\in\rho}P(r,s)\right\}.$$

So, in theory, this approach should give us the most accurate error bound. Notice that in this case Π_{med} is not stochastic.

3.1 Adaptive aggregation algorithm

For now we have been assuming that a specific state-space aggregation Φ of S was given. Let us now describe how to construct such partition for a DTMC having arbitrary structure of its state space. Also, in order to use adaptive aggregation as described in Algorithm 1, we need to specify a procedure for determining whether a partition is suitable for the current probability distribution.

First, we define adjacency of two states to be measured based on (mutual) transition probabilities: a pair of states that are connected by a significant transition probability are 'coupled' (in a probabilistic sense) and are good candidates to form a cluster. One would intuitively suggest this approach when treating a Markov chain as a directed weighted graph. Second, a requirement of cluster size to be inversely proportional to its transient probability will be fulfilled by performing the clustering in a bottom-up fashion, where we try to merge clusters together unless their resulting probability exceeds a given threshold δ. This way, states with significant probability mass (i.e. above threshold δ) will automatically form trivial clusters; conversely, states having negligible probability will be aggregated to large clusters. Combined with the definition of adjacency mentioned earlier, it will allow us to adequately partition any state space, regardless of its structure. The overall partitioning Algorithm 2 is presented below.

On line 1, we sort all transitions in descending order; this way we establish levels of adjacency between the states and then we favour those pairs of states connected by the most significant transitions by trying to aggregate them first. On lines 2 to 4 we construct a partition consisting of trivial clusters. We then proceed from the bottom to up and try to merge clusters together. First, we pick states *src* and *dst* that represent the source and the destination of a significant transition from T; then we find clusters ρ and σ containing the corresponding states and then merge them if their resulting probability sum will not exceed a specified threshold δ.

Algorithm 2: State space partitioning of a general DTMC

Input : DTMC (S, P), initial distribution p_0, probability threshold δ
Output: Partition Φ of S.

$T = \text{sort}(P)$; $\Phi = \emptyset$;
for $s \in S$ **do**
    $\Phi = \Phi \cup \{\{s\}\}$;
end for
for $(src, dst) \in T$ **do**
    $\rho = \text{clusterOf}(src)$; $\sigma = \text{clusterOf}(dst)$;
    if $\rho \neq \sigma$ *and* $\pi(\rho) + \pi(\sigma) < \delta$ **then**
        $\Phi = ((\Phi \setminus \{\rho\}) \setminus \{\sigma\}) \cup \{\rho \cup \sigma\}$;
        $\pi(\rho \cup \sigma) = \pi(\rho) + \pi(\sigma)$;
    end if
end for
return Φ;

Similarly as in Section 2.1.2, we will use several partitions that will adapt to the current probability distribution. The moment when a given partition is no longer considered to be appropriate is detected when any non-trivial cluster has accumulated a non-negligible amount of probability and hence starts to generate a significant error, i.e. procedure checkPartition() from Algorithm 1 asserts condition $\forall \sigma \in \Phi \ |\sigma| > 1 \Rightarrow \pi_k(\sigma) < \delta \cdot \delta'$, where δ' is some parameter. The choice of the parameters δ and δ' naturally allows us to adjust aggregation behaviour. For the choice of δ, big values will allow the clustering procedure to merge more states together, so larger values of δ usually mean larger state space reduction (and larger error). On the other hand, δ' is a parameter that controls how frequently reaggregations are performed: if $\delta' = 1$, partition is no longer appropriate as soon as some cluster ρ accumulates probability $\pi(\rho)$ larger than δ, and this can happen even after one iteration; if $\delta' > 1$, it gives a partition checker some sort of inertia in the sense that now a cluster can accumulate more probability without being considered inappropriate. So, larger values of δ' will result in less frequent reaggregations (and larger error). In both cases there is no guarantee on the maximum number of clusters or the minimum number of reaggregations. Notice that the choice of parameters gives rise to the conflict of efficiency versus precision: large δ allows to aggregate more and reduce complexity of the vector-matrix multiplications, for the price of increased error; similarly, large δ' allows us to recluster less frequently, again, for the price of increased error.

On a final note, let us agree that a more straightforward way of establishing a partition would be to minimize error in (3.6) while satisfying some additional constraints (e.g. a desired state space reduction). Although we studied general techniques for identifying clusters in directed graphs [20], none of them provided an elegant and efficient solution to the problem. Therefore, several heuristics described above were taken into account that gave rise to Algorithm 2. Also, note that in this algorithm we require $\pi(\rho)$ to satisfy a certain condition, not the product of $\pi(\rho) \cdot \tau(\rho)$ (which will ultimately generate an error). The reason for this is that constructing τ-terms on the fly is computationally demanding, so they are computed only after a partition is established. Analogously, during partition check, we assert whether $\pi(\rho)$ (not the corresponding product) satisfies a certain condition, because we have a guarantee that for $\delta' \geq 1$ our partition is valid before the first probability

propagation. However, in practice, asserting whether $\pi(\rho)$ or $\pi(\rho) \cdot \tau(\rho)$ satisfies a certain threshold does not lead to inherently different behaviour of the partition checker.

3.2 Experimental evaluation

Having now a broad arsenal of aggregation schemes and a complete algorithm they can be integrated into, it is time for experimental evaluation. Adaptive aggregation procedure accompanied by all three schemes (3.7)-(3.9), as well as DTMC analysis based on threshold abstraction, were implemented in PRISM, explicit engine; this engine does not use symbolic data structures for model construction and was proved to provide the best performance for general models of a moderate size (up to $\approx 10^7$ states). All of the experiments (including those in the next chapter) are run on a CentOS 6.5 server with 12x Intel Xeon E5-2640 (6 cores at 2.5 GHz) and 64 GB RAM with all the algorithms being executed sequentially (1 thread).

In our first set of experiments, we try to apply individual aggregation schemes in various scenarios on analysing a simple DTMC and compare the achieved accuracy, both empirical (comparing to exact result) and theoretical (upper error bound on the L_1-norm). We perform these experiments on two different models of different sizes exhibiting distinctive behaviour to eliminate any bias regarding the choice of the model under investigation. Overall, we perform three different experiments having various goals in mind:

- Experiment 1 (E1e, E1t): We evolve the model for 100 (exact) steps, then perform our first partitioning and compute abstract transition matrices using three different approaches: median-based (Med), based on average outgoing (Out) or incoming (In) transition probabilities. Then we perform a single step in this abstract setting. In all cases we will be using the same value of aggregation threshold and therefore each scheme will be working with exactly the same state space partition. We report empirical error (E1e) and theoretical error bound on the propagation error (E1t), i.e. $\|\mathbf{e_{101}}\|_1$ without aggregation error $\|\mathbf{e_{100}}\|_1$, which would be same in all cases since each technique uses the same state space partition[1]. The differences in the obtained values will arise only from of using different abstract transition matrices. The goal of this experiment is to compare one-step behaviour of the various abstract transition functions. For the case of average incoming probabilities, we will also compute theoretical bound using (2.7) (In') to check whether the new bound (3.6) (In) gives a better approximation.

- Experiment 2 (E2e, E2t): Same as E1e & E1t, but after aggregation we perform 100 consecutive steps without reclusterings. The goal of this experiment is to demonstrate the long-term behaviour of different aggregation approaches. Again, the value reported in E2t is $\|\mathbf{e_{200}}\|_1 - \|\mathbf{e_{100}}\|_1$, i.e. bound on the propagation error during each of the 100 steps after first aggregation.

- Experiment 3 (E3e, E3t): Same as E2e & E2t, but during 100 steps after the first aggregation we will perform 10 additional reclusterings at fixed times (105, 115, 125 etc.). Since for each of the schemas the probability distributions during the corresponding times will be approximately the same and the aggregation error is negligible

[1] We could also filter out aggregation error in E1e, although it will hardly make us any good: in this case errors are 'sign-sensitive' and can cancel each other out during the 101th step.

compared to propagation one, in E3t, again, we report only the bound on the propagation error. The goal of this experiment is to investigate behaviour of various approximations under regular reaggregations.

	E1e	**E1t**	**E2e**	**E2t**	**E3e**	**E3t**
Med	1.93E-23	9.77E-24	3.59E-4	3.71E-4	3.50E-4	3.50E-4
In'	2.07E-23	1.31E-20	3.58E-4	1.33E-1	2.01E-17	7.17E-14
In		1.28E-23		3.86E-4		2.81E-17
Out	2.51E-23	1.65E-23	6.40E-4	8.95E-4	2.85E-20	2.04E-19

Table 3.1: Accuracy of various aggregation schemas. Model: Lotka-Volterra [13], N = 400 (160k states), δ =1E-25 .

	E1e	**E1t**	**E2e**	**E2t**	**E3e**	**E3t**
Med	1.24E-9	8.09E-11	1.25E-7	1.40E-7	1.27E-7	1.27E-7
In'	1.26E-9	4.39E-6	1.21E-7	1.06E-2	1.02E-7	2.10E-4
In		1.07E-10		1.52E-7		1.11E-7
Out	1.29E-9	1.27E-10	1.94E-7	2.80E-7	1.68E-8	3.17E-8

Table 3.2: Accuracy of various aggregation schemas. Model: Prokaryotic Gene Expression [16], maxPop = 9 (700k states), δ =1E-10 .

The results for two different models of two different sizes are showed in tables 3.1 and 3.2, where the displayed values were truncated to two decimal places. First, from all experiments we unequivocally confirm that the new error bound (3.6) indeed gives several orders of magnitude better estimate than the one based on ϵ-terms (2.7). Second, experiment 1 shows us that median-based aggregation exhibits the best one-step behaviour, followed by incoming, followed by outgoing. Third, in experiment 2 we see a severe accuracy decrease in all of the methods: probability distribution has changed and in the absence of reclusterings we obtain a significant error. Fourth, although median-based aggregation performs slightly worse than incoming averaging, its theoretical bound of the actual error is still the best one. Finally, in experiment 3 we see that reclusterings can drastically improve the situation for incoming and outgoing approaches, with the latter having an edge of a couple of orders of magnitude.

The difference in the obtained values arises from how an individual scheme handles the problem of probability forwarding into big clusters. Median-based aggregation is very likely to pick $\underset{s\in\sigma}{\text{med}}\left\{\sum_{r\in\rho} P(r,s)\right\}$ equal to zero, because a majority of states in a big target cluster would be inaccessible in one step. Hence, no probability forwarding occurs at all, and the error is generated by the opposite effect: states that *are* accessible in one step will not get any probability at all. In the long run, it seems to be ineffective because abstract transitions equal to zero now decelerate the system and reaggregations cannot improve the situation; median-based aggregation, as expected, produces the best error bound of its actual error, yet this error is intrinsically poor and the scheme is of no great use to us. On the other hand, strategies based on averaging always propagate at least some probability mass, and the incoming version seems to be advantageous because a large size of a successor can

alleviate abstract transition probability and probability forwarding would be less apparent as compared to outgoing. The latter, however, seem to be much more susceptible to regular reclusterings.

In the next set of experiments, we put all the approximation techniques to the real test. We pick a concrete model, namely, Lotka-Volterra of 0.5M states and compute approximation of its transient probability distribution at time 10000. For a given precision (Acc), ranging from 1e-1 to 1e-5, each method is required to compute the result as fast as possible guaranteeing this precision; an accuracy of the method is computed using (3.6) for outgoing (Out) and incoming (In) averaging, using (2.7) for incoming (In') averaging and using probability loss for threshold abstraction (Tru). To ensure a fair comparison, parameters for each of the methods are tuned individually in each of the experiments in order to obtain the best computation time. Concrete values of parameters that yield the presented results as well as other statistics (total number of aggregations, state space reduction, etc.) can be found in `experiments.txt` file on the accompanying storage device. Results of this experiment are presented in Table 3.3 where we report acceleration with respect to the reference computation (10000 usual multiplications of matrices of size 0.5M). In this table we did not include data for median-based aggregation since for such large time horizon this approach failed to produce reasonable results. The choice of the model was completely arbitrary, and none of the methods could be more advantageous than other while analysing it, at least to our knowledge.

Acc	1E-1	1E-2	1E-3	1E-4	1E-5
Out	6.029	5.712	5.206	4.789	4.715
Tru	5.142	4.944	4.720	4.541	4.541
In	4.059	4.059	3.201	3.201	3.181
In'	3.380	2.935	2.785	2.785	2.650

Table 3.3: Maximum performance of various aggregation techniques. Model: Lotka-Volterra, N = 700 (.5M states), 10000 steps .

It is clear that Out > In > In'. Second inequality comes from the usage of better theoretical bounds, which means that incoming averaging that utilised τ instead of ϵ can make use of larger empirical error (by clustering more or reclustering less) to guarantee a certain precision. The first inequality is explained by the fact that outgoing averaging is more susceptible to reaggregations, and therefore fewer of those are required to guarantee a certain precision. We basically confirm our results from the first set of experiments. What is new here, however, is the illustration of behaviour of the state space truncation, which seems to be slightly inferior to outgoing averaging. Let us repeat this experiment with a different model, namely, a uniformised version of the two-component signalling pathway [25]; also, instead of transient analysis, both techniques will perform model checking.

State-space aggregation is a clear winner here. Let us also evaluate the precision of both methods similarly as in Table 3.1. We pick the same Lotka-Volterra model of size 160k from the first group of experiments and evaluate it using outgoing averaging (Out) and threshold abstraction (Tru). The strategy here is the following. First, we perform 100 exact steps, then we start approximating using the same aggregation/truncation threshold: with aggregation, states with probability below this threshold will be aggregated; with threshold abstraction, such states will be truncated. This way we perform 1, 100, 300, 500, 700 or 900 discrete steps more (in the case of aggregation, we also perform regular

Acc	1E-1	1E-2	1E-3	1E-4	1E-5
Out	9.191	8.373	7.308	5.149	4.179
Tru	7.389	5.596	4.819	4.014	3.109

Table 3.4: Maximum performance comparison. Model: two-component signalling pathway, population bounds [18,42]; property of interest is $\mathbb{P}(\Box^{\leq 10000} popRp \leq 27)$; number of states after PCTL driven transformation: 0.5M.

reclusterings, again at fixed times after 10 steps) and in each case we report empirical (e) and theoretical (t) error for both methods (for threshold abstraction both errors are defined as probability loss). The results are presented in Table 3.5. Keep in mind that, contrary to experiments from Tables 3.3 and 3.4, both aggregation and truncation use the same aggregation/truncation threshold.

Steps	101	200	400	600	800	1000
Tru	1.0E-8	5.2E-7	2.0E-6	4.1E-6	6.2E-6	7.9E-6
Out(e)	1.2E-8	2.1E-7	2.6E-7	3.1E-7	3.2E-7	2.1E-7
Out(t)	1.4E-8	2.3E-6	7.7E-6	1.5E-5	2.0E-5	2.4E-5

Table 3.5: Adaptive aggregation versus state space truncation accuracy comparison. Model: Lotka-Volterra, N = 400 (160k states), δ =1E-25.

It is clear that aggregation gives a better empirical error, which confirms our intuition that aggregating the state space and having at least an approximate idea where the residual probability is located is better than truncating it completely. On the other hand, threshold abstraction can provide an excellent theoretical bound on the error which ultimately beats approximation bound based on τ-terms. However, as Tables 3.3 and 3.4 suggest, this does not give threshold abstraction the necessary advantage: when allowed to tune parameters individually, outgoing averaging is capable of striking the perfect balance between state space reduction and a number of reclusterings in order to provide a more efficient approximation.

Chapter 4

Adaptive Aggregation for CTMCs

4.1 State-Space Aggregation for Standard Uniformisation

Having now an efficient aggregating method for the DTMCs, let us now combine it with uniformisation technique in order to analyse continuous-time chains. In AU, transition probability matrix for the internal discrete process changes each iteration with varying uniformisation rate; in SU, however, this rate is fixed and the transition probability matrix is computed only once, so there seems to exist a way we could aggregate it and perform computation in the abstract setting. Such approach was already presented in [1], although a rigorous error bound is yet to be derived, which is the main interest of this section.

Recall that SU works by constructing a uniformised DTMC from the rate matrix using a single uniformisation rate; it proceeds by computing transient probabilities for this DTMC and then weighs them using a Poisson distribution $\psi_{qt}(\cdot)$. The Fox-Glynn algorithm provides us bounds L, R that allow us to truncate the infinite sum and compute the overall result as in (2.12). The error associated with this truncation is the probability loss $\|\mathbf{p_t}\|_1 - \|\hat{\mathbf{p}}_\mathbf{t}\|_1 = \sum_{s\in S} p_t(s) - \sum_{s\in S} \hat{p}_t(s) = 1 - \sum_{s\in S} \hat{p}_t(s)$.

Now let us approximate $\hat{p}_t$ with $\tilde{\hat{p}}_t$ by replacing u_k with approximations $\tilde{u}_k$ in (2.12). Each $\tilde{u}_k$ has an uncertainty e_k associated with it and each $\tilde{u}_k$ is being weighted with $\psi_{qt}(k)$, so the overall error contributed is $\sum_{k=L}^{R} \|\mathbf{e_k}\|_1 \cdot \psi_{qt}(k)$. We also lose some probability mass during the sum truncation and to estimate this loss we use the following lemma.

Lemma 2. Let $\{\mathbf{v_k}\}_{k\in\mathbb{N}_0}$ be an infinite sequence of vectors of the same dimension and let $\mathbf{v} \coloneqq \sum_{k=0}^{\infty} \mathbf{v_k} \cdot w_k$, where w_k are non-negative scalars for which $\sum_{k=0}^{\infty} w_k = 1$. If norms of $\mathbf{v_k}$ are bounded by some v^*, i.e. $\exists v^*\ \forall k \in \mathbb{N}_0\ \|\mathbf{v_k}\|_1 \leq v^*$, then $\|\mathbf{v}\|_1 \leq v^*$.

Proof.

$$\|\mathbf{v}\|_1 = \left\| \sum_{k=0}^{\infty} \mathbf{v_k} \cdot w_k \right\| = \sum_{k=0}^{\infty} w_k \cdot \|\mathbf{v_k}\|_1 \leq \sum_{k=0}^{\infty} w_k \cdot v^* = v^* \sum_{k=0}^{\infty} w_k = v^*.$$

□

Corollary 2.1. Let $\hat{\mathbf{v}} \coloneqq \sum_{k=L}^{R} \mathbf{v_k} \cdot w_k$ be a truncation of $\mathbf{v}$ for arbitrary bounds L, R, where $\{\mathbf{v_k}\}$ is consistent with Lemma 2. Then for the total mass lost after this truncation it holds:

$$\|\mathbf{v}\|_1 - \|\hat{\mathbf{v}}\|_1 \leq v^* - \|\hat{\mathbf{v}}\|_1. \tag{4.1}$$

If we treat symbols above in the context of CTMCs, $\mathbf{v_k}$ of course refers to $\mathbf{u_k}$ (or $\tilde{\mathbf{u}}_\mathbf{k}$), $\mathbf{v}$ refers to $\mathbf{p_t}$ (or $\tilde{\mathbf{p}}_\mathbf{t}$), $\hat{\mathbf{v}}$ refers to $\hat{\mathbf{p}}_\mathbf{t}$ (or $\tilde{\hat{\mathbf{p}}}_\mathbf{t}$) and w_k is analogous to $\psi_{qt}(k)$ (or $\beta(k)$). For SU, AU, FSU or FAU, each of the $\|\mathbf{v_k}\|_1$ is bounded by 1 and so (4.1) takes its usual form $1 - \|\hat{\mathbf{v}}\|_1$. For the state-space aggregation, there is in general no guarantee on the form of approximations $\mathbf{v_k}$, see the following example.

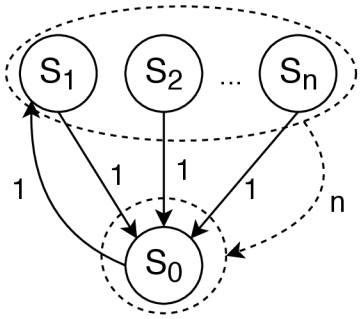

Figure 4.1: Aggregation strategy based on average incoming probabilities may lead to diverging transient probabilities.

Example 6. Consider a DTMC in Figure 4.1 and assume that it starts at s_1. We are free to use different partitions as time progresses, so let us assume that we will use trivial partition (i.e. unaggregated model) during even iterations and a partition $\Phi_{odd} = \{\{s_0\}, \{s_1, \ldots, s_n\}\}$ during odd ones. Therefore, during the first iteration, a probability 1 will get transported using abstract transition probability n, resulting in $\tilde{p}_1(s_0) = \pi_1(s_0) = n$. During the next step, this n is transported back to s_1. Again, apply Φ_{odd} before propagating to obtain $\tilde{p}_3(s_0) = \pi_3(s_0) = n^2$. It is clear that in this scenario the total probability mass in the system will be increasing exponentially. Although this example is somewhat artificial, it conveys a message that for arbitrary aggregation approaches there is no theoretical guarantee on the form of resulting approximations.

The conclusion is: in order to use aggregation techniques in combination with the Fox-Glynn scheme, it is crucial to appeal to those abstractions for which $\{\mathbf{v_k}\}$ is consistent with Lemma 2. A perfect candidate here is the outgoing averaging approach that preserves stochasticity, $\|\mathbf{v_k}\|_1 = 1$, and so probability loss can be computed analogously to SU/AU. As a bonus, this technique proved to be the most efficient one for the analysis of discrete-time chains. Henceforth, under the term 'state-space aggregation' we will implicitly assume the technique based on average outgoing probabilities.

The rest of the algorithm remains conceptually the same: we work with the abstraction of the uniformised DTMC, where notions of state adjacency, aggregation checking, etc. are analogous to those developed in the previous chapter. We are interested in computing approximate transient probability distributions of this DTMC that are then being weighed by Poisson probabilities. The overall error is the sum of the probability loss and approximation error for uniformised DTMC:

$$\|\mathbf{e_t}\|_1 \leq 1 - \|\tilde{\hat{\mathbf{p}}}_\mathbf{t}\|_1 + \sum_{k=L}^{R} \|\mathbf{e_k}\|_1 \cdot \psi_{qt}(k). \tag{4.2}$$

A combination of the state-space aggregation with SU will be denoted as SU+.

4.2 State-Space Aggregation for Adaptive Uniformisation

We have seen previously how adaptive uniformisation can drastically shorten the required number of iterations without any precision penalty. The goal of this section is to adopt this approach to the state space aggregation. We cannot apply adaptive uniformisation directly since we have developed our aggregation scheme for (uniformised) DTMCs: we would have to continuously recompute its transition probability matrix each time a uniformisation rate q_i changes, which is completely impractical. Instead, we could define this matrix as a function of the uniformisation rate.

Let (S, R) be CTMC and Q be the infinitesimal generator associated with R. Assume a specific state space aggregation Φ of S is given. Using outgoing averaging (3.8), we have

$$\Pi(\rho, \sigma) = \frac{1}{|\rho|} \sum_{r \in \rho} \sum_{s \in \sigma} unif_R^q(r, s), \tag{4.3}$$

where q is the uniformisation rate. In the case where $\rho \neq \sigma$, we have $\forall r \in \rho \; \forall s \in \sigma \; r \neq s$, so that $unif_R^q(r, s) = \frac{Q(r,s)}{q}$ and therefore equation (4.3) becomes

$$\Pi(\rho, \sigma) = \frac{1}{|\rho|} \sum_{r \in \rho} \sum_{s \in \sigma} \frac{Q(r, s)}{q} = \frac{1}{q} \cdot \frac{1}{|\rho|} \sum_{r \in \rho} \sum_{s \in \sigma} Q(r, s).$$

On the other hand, if $\rho = \sigma$, we have to keep in mind self-loops:

$$\begin{aligned} \Pi(\rho, \rho) &= \frac{1}{|\rho|} \sum_{r \in \rho} \left[1 + \frac{Q(r, r)}{q} + \sum_{s \in \rho, r \neq s} \frac{Q(r, s)}{q} \right] = \frac{1}{|\rho|} \sum_{r \in \rho} \left[1 + \sum_{s \in \rho} \frac{Q(r, s)}{q} \right] \\ &= \frac{1}{|\rho|} \left[\sum_{r \in \rho} 1 + \sum_{r \in \rho} \sum_{s \in \rho} \frac{Q(r, s)}{q} \right] = 1 + \frac{1}{q} \cdot \frac{1}{|\rho|} \sum_{r \in \rho} \sum_{s \in \rho} Q(r, s). \end{aligned}$$

We arrive at the following definitions:

Definition 10. Let (S, R) be CTMC with infinitesimal generator Q and let Φ be the partition of S. *An abstract infinitesimal generator* $\Theta : \Phi \to \Phi$ is the function defined as

$$\Theta(\rho, \sigma) = \frac{1}{|\rho|} \sum_{r \in \rho} \sum_{s \in \rho} Q(r, s).$$

Definition 11. Let (S, R) be CTMC with partition Φ of S. *The exit rate of the abstract state* σ is the maximum exit rate of states within this cluster:

$$E(\sigma) \coloneqq \max_{s \in \sigma} E(s).$$

Proposition 3. Let (S, R) be CTMC, Φ be the partition of S, Θ be the corresponding abstract infinitesimal generator and $q \geq \max_{\sigma \in \Phi} E(\sigma)$ be the uniformisation rate. Then

$$\Pi_{out}(\rho, \sigma) = \begin{cases} 1 + \frac{\Theta(\rho,\sigma)}{q}, & \text{if } \rho = \sigma \\ \frac{\Theta(\rho,\sigma)}{q}, & \text{otherwise.} \end{cases}$$

Its proof was demonstrated in the derivation above.

Corollary 3.1. For the error factors $\tau(\cdot)$ associated with partition Φ and abstract transition function $\Pi(\cdot,\cdot)$ based on average outgoing probabilities, it holds:

$$\tau(\rho,\sigma) = \frac{1}{q}\sum_{s\in\sigma}\left|\frac{\Theta(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} Q(r,s)\right|.$$

Proof.

Case $\rho \neq \sigma$:

$$\begin{aligned}\tau(\rho,\sigma) &= \sum_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} unif_R^q(r,s)\right| = \sum_{s\in\sigma}\left|\frac{1}{|\sigma|}\frac{\Theta(\rho,\sigma)}{q} - \frac{1}{|\rho|}\sum_{r\in\rho}\frac{Q(r,s)}{q}\right| \\ &= \frac{1}{q}\sum_{s\in\sigma}\left|\frac{\Theta(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} Q(r,s)\right|.\end{aligned}$$

Case $\rho = \sigma$:

$$\begin{aligned}\tau(\rho,\sigma) &= \sum_{s\in\sigma}\left|\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} unif_R^q(r,s)\right| \\ &= \sum_{s\in\sigma}\left|\frac{1}{|\sigma|}\left[1+\frac{\Theta(\rho,\sigma)}{q}\right] - \frac{1}{|\rho|}\left[1+\frac{Q(r,r)}{q}+\sum_{s\in\sigma,r\neq s}\frac{Q(r,s)}{q}\right]\right| \\ &= \sum_{s\in\sigma}\left|\frac{1}{|\sigma|}+\frac{1}{|\sigma|}\frac{\Theta(\rho,\sigma)}{q} - \frac{1}{|\rho|} - \frac{1}{|\rho|}\sum_{r\in\rho}\frac{Q(r,s)}{q}\right| \\ &= \frac{1}{q}\sum_{s\in\sigma}\left|\frac{\Theta(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} Q(r,s)\right|.\end{aligned}$$

□

Algebraic manipulations above allowed us to express $\Pi(\cdot,\cdot)$ and $\tau(\cdot)$ in terms of the uniformisation rate q. In the case when this rate varies during each iteration, we are now able to construct $\Pi(\cdot,\cdot)$ for probability propagation, as well as $\tau(\cdot)$ for error estimation, on the fly. We will no longer work with P and its abstraction Π, but with Q and its abstraction Θ: structure (S,Θ) will be therefore referred to as *an abstract CTMC*. Using the definition of the uniformisation rate (2.13) along with the Definition 11 of the exit rate of a cluster, we can compute current uniformisation rate q_i as

$$q_i \geq \max_{s\in S}\{E(s) \mid \tilde{u}_i(s) > 0\} = \max_{\sigma\in\Phi}\{E(\sigma) \mid \pi_i(\sigma) > 0\}.$$

The resulting method combines both state space aggregation and adaptive uniformisation and its procedure can be outlined as follows. First, we construct a state space partition Φ of S using initial probability distribution and aggregation threshold δ_{agg}, where adjacency of two states is now defined in terms of the rate matrix R. The resulting partition would be exactly the same if we uniformised R first: uniformisation rate q does not affect relative magnitudes of between-state transitions. Having Φ, we can construct abstract infinitesimal generator Θ, along with the value of

$$\sum_{\sigma\in\Phi}\sum_{s\in\sigma}\left|\frac{\Theta(\rho,\sigma)}{|\sigma|}-\frac{1}{|\rho|}\sum_{r\in\rho}Q(r,s)\right|$$

for each $\rho \in \Phi$, which, according to Proposition 3 and Corollary 3.1, will allow us propagate probability of the uniformised DTMC and estimate the propagation error, regardless of the current uniformisation rate q_i. This rate is given by the largest exit rate within the set of active states, which, according to Definition 11, is equivalent to the largest exit rate within the set of active clusters. The error associated with this approximation can be estimated using (4.2). Finally, we can combine the method above with the threshold abstraction where in each iteration we truncate insignificant clusters, i.e. those with probability smaller than δ_{tru}, in order to obtain even smaller uniformisation rates. The resulting technique, denoted FAU+, is a hybrid between the state-space aggregation and FAU. Aggregation threshold δ_{agg} that defines how much of the probability mass can constitute a cluster, along with the truncation threshold δ_{tru} defining which clusters are to be considered insignificant, will drive the overall behaviour of the method, where in the limiting cases we can even simulate other approximation techniques, see Table 4.1, where AU+ denotes a combination of AU with the state-space aggregation. Adaptivity of FAU+ will allow us to perform fewer iterations as compared to SU+.

δ_{agg}	δ_{tru}	Simulated method
0	0	AU
0	> 0	FAU
> 0	0	AU+
> 0	> 0	FAU+

Table 4.1: FAU+ parameters influence the overall behaviour of the method.

4.3 Experimental evaluation

We wish to compare both uniformisation techniques utilising threshold abstraction (FSU & FAU) and combinations of SU & FAU with the state-space aggregation (SU+ & FAU+). Accuracy comparison for aggregation versus truncation from Table 3.5 automatically translates to SU+ versus FSU since now we compute the same transient probabilities and only weigh them with Poisson probabilities afterwards. Accuracy comparison involving FAU or FAU+ is tricky because of varying uniformisation rates that lead to the non-linear progression of time. However, AU differs from SU only in the total number of iterations and not in the overall precision, so general conclusions from Table 3.5 can also be applied when comparing SU+/FAU+ with FAU.

Therefore, we proceed by reconstructing experiments regarding the overall performance of individual techniques from Tables 3.3 and 3.4 in the continuous setting. For the first experiment we choose to model check the signalling pathway model of exactly the same size as in 3.4. In table 4.2, for each of the techniques we report the time acceleration (with respect to SU) to guarantee a certain accuracy (Acc). Concrete values of parameters that yield the presented results as well as other statistics (total number of aggregations, state space reduction, etc.) can be found in `experiments.txt` file on the accompanying storage device.

Acc	1e-1	1e-2	1e-3	1e-4	1e-5
SU+	3.4013	2.7783	2.3618	2.0365	1.6762
FAU+	4.9886	3.6509	3.0145	2.4026	2.1696
FSU	3.8363	2.7357	2.1929	1.9260	1.6473
FAU	5.3504	3.9517	3.1911	2.6817	2.3142

Table 4.2: Performance of various approximation techniques. Model: two-component signalling pathway, population bounds [18,42]; property of interest is $\mathbb{P}(\Box^{\leq 3} popRp \leq 27)$; number of states after CSL driven transformation is 0.5M, upper FG bound is 6131.

First, we observe that SU+ and FSU exhibit approximately the same behaviour. Contrary to what one could have deduced from Table 3.4, now both techniques are dealing with a CTMC that is being uniformised with a much lower rate and therefore the inner DTMC progresses much faster. State space truncation, because of the nature of its algorithm, cannot notice this difference, but aggregation now has to perform more reclusterings. Second, we see that adaptive approach employed in both FAU and FAU+ leads to a considerable performance improvement: in this case FAU/FAU+ perform roughly twice as less iterations as FSU/SU+ and are significantly faster than their non-adaptive counterparts. Notice that FAU+ performs better than SU+ since it is able to simulate aggregation/truncation strategy close to that of FAU, and therefore both FAU+ and FAU exhibit a similar behaviour.

In the second experiment we choose the Lotka-Volterra model of exactly the same size and a continuous time horizon t for which the Fox-Glynn scheme would give us an upper bound R to be approximately 10000, as in 3.3. Results are illustrated in Table 4.3.

Acc	1e-1	1e-2	1e-3	1e-4	1e-5
SU+	7.2664	6.8700	6.1350	6.2728	5.9660
FAU+	5.9984	5.9984	5.9984	5.6275	5.1405
FSU	4.2299	4.2235	4.1854	4.1069	4.0239
FAU	4.8737	4.6581	4.5796	4.5824	4.3070

Table 4.3: Performance of various approximation techniques. Model: Lotka-Volterra, N = 700, 0.5M states; t = 0.5, upper FG bound is 10594.

First, observe that, although FAU performs fewer iterations (again, approximately twice as less) than its non-adaptive counterpart, it does not seem to get such a drastic advantage over FSU. The difference from the previous experiment is that now truncation techniques FSU/FAU can achieve a much greater state space reduction (200x over 20x in 4.2); probability propagation is now computationally less demanding, and therefore reducing the overall number of iterations for the price of solving a birth process during each step cannot help FAU much, although it still slightly outperforms FSU.

Second, SU+ cleary surpasses both FSU and FAU. The difference from the previous experiment again arises from the state space reduction improvement and is explained from the algorithmic standpoint as follows. Both aggregating and truncating techniques gain performance increase when dealing with smaller state spaces, however, in this concrete example, FSU/FAU gained much less because of the way the probability propagation phase was implemented. The issue here is that the implementation of truncation techniques (Tru, FSU,

FAU, FAU+) we are currently evaluating propagate probability blindly to a preallocated array, without remembering concrete states that were 'discovered' during this propagation; such states are identified before the next iteration when constructing a set of active states, which is achieved by scanning over the whole state space. The second approach, probably more intuitive one, would be to remember discovered states during propagation phase, i.e. to collect target states to e.g. a tree set or a hash set, and then only scan through this set and remove insignificant ones before successive propagation. It might seem that the second approach would be superior since we are not dealing with the whole state space. In practice, however, it turns out that the opposite is true. When the state space is large and, in particular, the set of *active* states is large, maintaining a collection of discovered states during the propagation phase is much more computationally demanding than propagating the probability blindly and then scanning over the whole state space once before the next iteration. In other words, both implementations will lead to performance improvement when dealing with smaller state spaces, but the second approach can gain more from larger reductions. Hence, if we implemented the second variant, FSU/FAU would unquestionably perform much better in 4.3, but, analogously, *truncation would perform much worse in 4.2 and SU+ would become superior*. Both implementations were tested, but the first variant is preferred because it can handle larger state spaces. This example was specifically designed to illustrate this property and to pinpoint that, regardless of the implementation, state-space aggregation is capable of outperforming existing truncation-based techniques. The subtlety of implementation plays a great role when comparing efficiency of individual techniques and it will be marginally revisited in the next chapter. As to the FAU+, we see that now it performs worse than SU+, which is explained using the exact same argument above. Overall, FAU+ could not provide a balance between aggregation and truncation and could not outperform the best of SU+/FAU alone. However, when faced with a model of the unknown nature, where applicability of SU+ versus FAU is under question, a diverse behaviour of FAU+ can considerably boost the analysis.

From the experiments above we arrive at several conclusions. First and foremost, we have succeeded to establish a state space aggregation technique that can provide an adequate and efficient approximation of the CTMC analysis, both transient and model checking. Second, for some classes of models and regardless of the implementation of techniques of interest, state space aggregation even outperforms existing methods, including FAU. Finally, we managed to integrate aggregation with adaptivity to reduce the required number of iterations, which may also turn out to be advantageous.

Chapter 5

Final Considerations

The first part of this chapter describes the implementation of all techniques discussed earlier within the PRISM framework. The second part is a collection of ideas, conjectures or just informal thoughts that do not directly contribute to any of the previous chapters, yet might be helpful to those willing to continue the research of approximative techniques. The closing part is the conclusion.

5.1 Implementation

As was previously mentioned, all of the methods and aggregation schemes were implemented for PRISM [17], explicit engine, in Java language. Although PRISM has general procedures like SU or FAU already implemented, we had to rewrite them into a single package in order to ensure a completely fair comparison and to simplify the setting up of the experiments and the collection of statistics. The resulting package `aggregation` contains all the procedures and can be found at `src/explicit/`. Some other files outside the package were slightly modified, sometimes to fix minor bugs, but mostly to redirect the computation flow into `aggregation.AggregationModelChecker`. This module serves as a playground where one can set up the experiments, evaluate them and construct reports; it also provides explicit procedures for constructing infinitesimal generators, uniformised DTMCs or PCTL/CSL driven transformations.

Let us cover other modules in this package from the bottom to up.

- `Propagable` is a wrapper over CSR (row start, column, data) representation of sparse transition probability/rate matrices; it allows collecting pairs of states connected by a significant transition, computing exit rates, probability propagation and state space truncation. `PropagableAbstract` is its abstract[1] extension that quantifies error during the probability propagation and provides procedures for vector (de-)aggregation.

- `Outgoing`, `Incoming` and `Median` are concrete realisations of `PropagableAbstract` that construct the abstract matrices using the corresponding scheme; the code of these modules could be refactored, but, since most of the time we were interested in the outgoing averaging, we found it unnecessary to overload it with flags and conditions. Also, `Incoming` module contains a flag that switches between ϵ and τ as error factors.

[1] in both senses: it operates in aggregated setting and it cannot be instantiated

- `Cluster` and `Partition` provide abstractions that allow to work will collections of states (clusters) and collections of clusters (partitions). Of particular interest might be the `reconfigure()` procedure that implements the bottom-up partitioning algorithm with the constant-time cluster merging (via linked lists) and almost constant-time cluster search (via dynamic tree-like structure), resembling the disjoint-set data structure [11].

- `BirthProcess` is a birth process solver; its instance can be sequentially supplied with rates $q_0, q_1, \ldots$ to obtain the corresponding $\beta_0, \beta_1, \ldots$.

- `Solver` is an interface, namely, a template (algorithm) for Markov chain transient analysers that do not perform aggregation; `SolverAbstract` is its aggregating counterpart. Both interfaces provide a default `run()` method that controls the behaviour of analysis and the collection of statistics in a uniform way.

- Finally, there are a total of eight distinct modules that implement those interfaces. For non-aggregating one, there are:

  - `Propagation` – DTMC classic analysis;
  - `Truncation` – threshold abstraction for DTMCs;
  - `StandardUniformisation` – SU;
  - `FastStandardUniformisation` – FSU;
  - `AdaptiveUniformisation` – FAU.

  Aggregating ones include:

  - `PropagationAbtract` – DTMC aggregating analysis;
  - `StandardUniformisationAbstract` – SU+;
  - `AdaptiveUniformisationAbstract` – FAU+.

The parameters to each of those methods are supplied during the experiment setup in `aggregation.AggregationModelChecker` module. Note that one could introduce an inheritance and derive each of the techniques as an extension (or, conversely, a generalisation) of another; this approach was actually implemented and tested, but turned out to be inefficient due to repetitive calls of polymorphic functions, plus it was not trivial to introduce a new technique. The resulting architecture presented here can be easily modified and provides a fair method comparison due to the shared `run()` environment in `Solver` or `SolverAbstract` interfaces.

Additionally, three major MATLAB scripts were written:

- `plot_evolution.m` allows to visualise projections of probability distributions to concrete values of variables, over a range of discrete steps. Refer to the script and to the `aggregation.AggregationModelChecker::dtmcExport` procedure for usage description.

- `plot_partition.m` allows to visualise state space partitions. Refer to the script and to the `aggregation.AggregationModelChecker::partitionExport` procedure for usage description.

- `fau.m` is a MATLAB implementation of FAU; the Fox-Glynn scheme is not used, instead, distribution starts at $L = 0$ and Poisson probabilities are generated until a given precision is satisfied.

The source code along with further details regarding installation, experiment setup and measured data can be found on the accompanying storage device.

5.2 Further Research

At the end of Chapter 4 we mentioned an impact of the implementation on the overall performance of the method. First, we have seen that distinct implementations of FAU perform differently under various models: one might consider designing FAU capable of switching between these implementations on the fly in order to achieve better performance. Similarly, for a given model, FAU+ is capable of demonstrating a broad behaviour range under various parameter values: learning how to adaptively tune these parameters will definitely lead to the overall performance improvement. Finally, there are numerous procedures in the state-of-the-art aggregation algorithm that should be studied. In particular, reclustering (or, rather, clustering) of a chain is an extremely expensive operation that can add up to 50% of the overall computation time, so the efficient design of this algorithm can significantly increase performance of the method. Furthermore, the clustering procedure covers the partitioning of the state space (see `aggregation.Partition`) and constructing the abstract transition matrix, as well as error factors (see `aggregation.Outgoing`) and there might be a way to interleave both operations. Constructing error factors along with clustering can drive the clustering algorithm and one can arrive at more suitable partitions: even slight changes in the resulting clustering can have an enormous impact on the overall precision of the method. For instance, we found out that sorting the transitions (see Algorithm 2, line 1) and favour pairs of states connected by a larger transition can lead to precision improvement of up to two orders of magnitude, as compared to the case where we take any two states connected by a non-zero transition. At the same time, sorting is a rather cheap operation that is performed only once before the first aggregation.

Moreover, we studied general approaches for the clustering of directed weighted graphs, but as was mentioned earlier, we have had no success. The ideal procedure must be kept simple since frequent reaggregations are the key to acceptable precision. We also analysed various clustering techniques experimentally and examined their effect on the empirical and theoretical error; one particular approach exhibited a peculiar behaviour and is worth discussing here.

Consider a simple Lotka-Volterra model; its state space is encoded using two variables – number of predators vs. number of prey – and can be visualised in a two-dimensional plane. Suppose we have progressed the model for some time and now we want to perform its first aggregation. We look for a partition that would minimise generated error, say, for the next 50 iterations, after which we will reaggregate the chain. A state-of-the-art algorithm produces clustering depicted in Figure 5.1 on the left. We can see that a majority of the probability mass is concentrated around (15,40), where active states form trivial clusters; far from this point we effectively have zero probability and therefore we use a single big orange cluster, this cluster will be referred to as *slave*; in between, we use clusters of the medium size[2]. Let us compute error factors $\tau(\cdot)$ for each of the cluster; their magnitude is

[2] A strange bar-shape of clusters of medium size reflects the underlying structure of the state space. Transitions from any state can occur up (a prey is born), to the left (a predator dies) and to the lower

illustrated in Figure 5.2 on the left, where warm colors correspond to higher values of error factor (concrete colormap is not depicted). We can identify four groups of states:

1. Trivial clusters inside the set of active states. A cluster in this group has all of its successors trivial and generates no error at all.

2. Trivial clusters on the border of the set of active states. This cluster has some successors of medium size and produces some small error.

3. Clusters of medium size. A cluster in this group has a gigantic successor (slave) and produces significant error.

4. Slave cluster. Has successors of medium size and one large successor (itself); moderate value of $\tau(\cdot)$ multiplied by insignificant transient probability yield a negligible error.

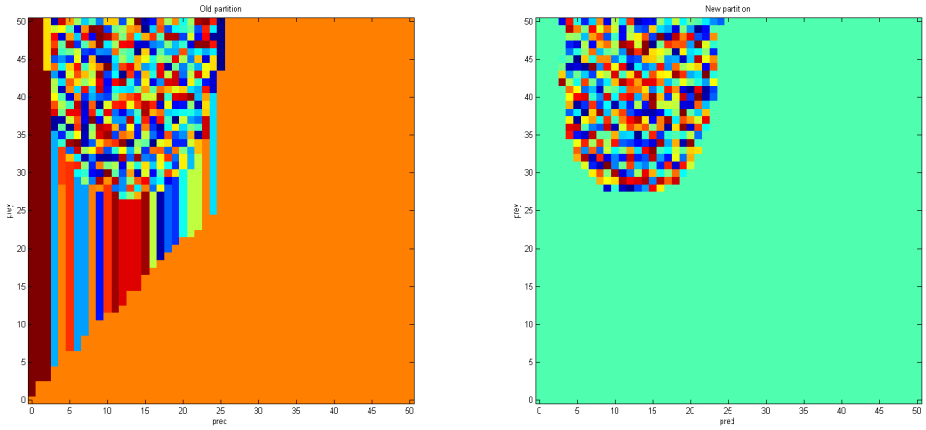

Figure 5.1: A state space partition using Algorithm 2 (left) and by simulating a truncation (right).

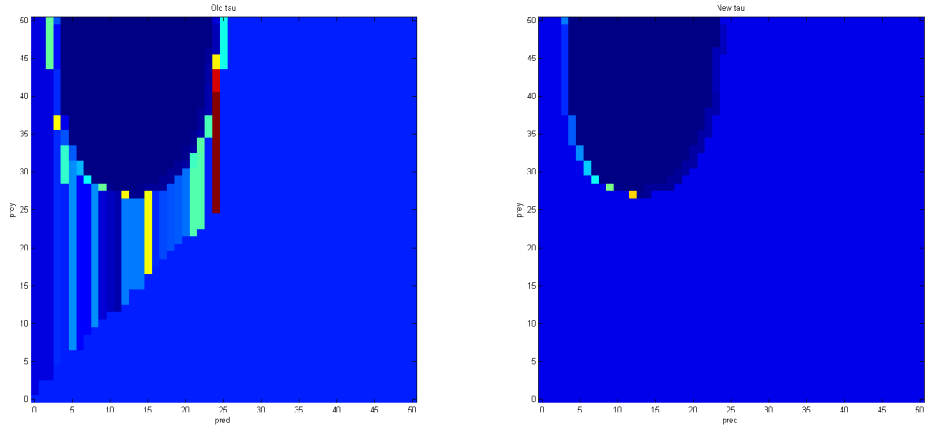

Figure 5.2: A relative magnitude of error factors for the corresponding partition.

right (a predator is born and a prey dies). In the upper-left side of the state space, where we have a lot of prey and not so few of predators, a prey is more likely to be born, and so transitions up are favored during partitioning.

The obtained distribution illustrates the following fact: $\tau(\rho, \sigma)$ *is large when successor* σ *is large* (notice that $\tau(\rho, \sigma) = 0$ when $|\sigma| = 1$) and this error factor captures probability forwarding *into* the cluster. At the same time, slave cluster has a big successor (itself), yet its $\tau(\cdot)$ is not as large as for medium clusters. The reason for this is that slave cluster has a high *intra-cluster density* [20]: a majority of states that constitute the slave would propagate probability to this same cluster; hence, no error is introduced when we replace these concrete transitions by a high abstract self-loop ($\Pi(slave, slave) \approx 1$). Multiplied by the small value of $\pi(slave)$, this big cluster would generate little to no error. We arrive at the conclusion that big clusters are 'bad' not because they produce significant error, but because this error is generated *because of them*.

Equipped with this knowledge, let us find an 'ideal' clustering. It turns out that the best candidate that minimises the distribution of error factor is the one illustrated in Figure 5.1 on the right. This clustering consists of trivial clusters as a set of active states (those with probability above the aggregation threshold) and a single slave cluster. The right side of Figure 5.2 depicts a distribution of $\tau(\cdot)$, where colors are consistent with those on left. We see that, again, clusters inside the set of active states produce no error; clusters on its border produce a slightly larger error (their successor is now large); the slave cluster now has a smaller value of $\tau(\cdot)$ since now it is even more dense and its other successors are all trivial. Most importantly, we eliminated a pass of medium-size clusters that were completely inappropriate before.

Is this the clustering we are looking for? No. Although it indeed minimises the error factor and produces a small amount of error during the next 10-20 iterations, this partition is impractical in the long run, when a shift of probability distribution occurs and states on the boundary of the set of active states start to forward significant probability mass into the slave. It would help to recluster the state space frequently; ideally, we could refine the set of active states after each iteration and reconstruct slave cluster consisting of inactive states. This aggregating approach obviously resembles behaviour of the state space truncation, except for the fact that threshold abstraction propagates only into concrete (maybe undiscovered) states and never forwards a probability.

So, ideal aggregation strategy would be the one that simulates threshold abstraction, yet this conclusion is inconsistent with results we obtained in experiments 3.3 and 3.4, where we proved that 'normal' clustering is the superior one. The paradox here is that we have been answering the wrong question: we attempted to minimise *theoretical* error associated with our aggregation strategy. Although theoretical error is crucial, it is the only measurement of the precision of the method and a good bound can preserve several orders of magnitude of accuracy, it does not seem to help us with finding the best aggregation strategy. We fell into the same trap when we derived median-based aggregation scheme: although it indeed produces the best theoretical bound of its empirical error, this approach is intrinsically inaccurate and impractical. The exact same argument applies to the truncation-like aggregation. On the other hand, when we had attempted to improve *empirical* accuracy by employing the aggregation scheme that preserves stochasticity, or by sorting significant transitions before the first aggregation, we had always been successful.

In the previous text we tried to quantify the error vector associated with our aggregations. In theory, this error can be tracked precisely in order to obtain exact results, but we want to reduce the overall complexity of this estimation. Therefore, we never worked with the vector directly, but with its L_1-norm, a 1-step update of which was proved to be computationally equivalent to performing a scalar product of two vectors over the abstract state space, as in (3.6). However, L_1 is not the only metric that can be imposed on a vector,

others include L_∞-norm (maximum deviation) or even point-wise error estimates $\varepsilon_k(s)$ of $e_k(s)$ for concrete states[3]. Unfortunately, despite numerous attempts, we never succeeded to derive meaningful and practical bounds on anything beside L_1, where we managed to capture a magnitude (i.e. sign-insensitive value) of the error generated in each cluster during each step. Any estimate of the error vector that allows to quantify a point-wise error will be extremely beneficial for model checking algorithms where the quantity of interest is often computed based on a transient probability of a single state. Truncation methods rely solely on the probability loss that serves as an estimate of L_1-norm and its doubtful that it can be improved. On the other hand, aggregation methods are potentially capable of estimating any type of error since, as comparison of the empirical error suggests, aggregating the state space preserves information about the probability distribution to a higher degree. Deriving a bound on anything better than L_1-norm will allow to overcome the theoretical accuracy gap between aggregation and truncation methods. Furthermore, introducing a sign-sensitive error tracking might be advantageous since positive and negative error accruals can cancel each other out.

Let us present a final observation concerning the estimate of the error vector. Let $s \in \sigma$. Applying equation (3.3) in the aggregated setting, we obtain

$$\begin{aligned} e_k(s) &= \sum_{r\in S} e_{k-1}(r)P(r,s) + \sum_{r\in S} \tilde{p}_{k-1}(r)\left(\tilde{P}(r,s) - P(r,s)\right) \\ &= \sum_{r\in S} e_{k-1}(r)P(r,s) + \sum_{\rho\in\Phi}\sum_{r\in\rho} \frac{\pi_{k-1}(\rho)}{|\rho|}\left(\frac{\Pi(\rho,\sigma)}{|\sigma|} - P(r,s)\right) \\ &= \sum_{r\in S} e_{k-1}(r)P(r,s) + \sum_{\rho\in\Phi} \pi_{k-1} \sum_{r\in\rho}\left(\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} P(r,s)\right). \end{aligned}$$

Let

$$\gamma(\rho) \coloneqq \sum_{r\in\rho}\left(\frac{\Pi(\rho,\sigma)}{|\sigma|} - \frac{1}{|\rho|}\sum_{r\in\rho} P(r,s)\right)$$

be a 'signed' version of $\tau(\rho)$. If $\mathbf{e_k}$ and $\pi_\mathbf{k}$ are row vectors and $\gamma \coloneqq [\gamma(\rho)]_{\rho\in\Phi}$ is a column vector, than equation above can be expressed as:

$$\mathbf{e_k} = \mathbf{e_{k-1}} \cdot \mathbf{P} + \pi_\mathbf{k-1} \cdot \gamma.$$

Let us now expand this recursion:

[3]Note that $e_k(s) \le \varepsilon_k(s) \le \|\mathbf{e_k}\|_\infty \le \|\mathbf{e_k}\|_1$.

$$
\begin{aligned}
\mathbf{e_1} &= \mathbf{e_0} \cdot \mathbf{P} + \pi_\mathbf{0} \cdot \gamma, \\
\mathbf{e_2} &= \mathbf{e_1} \cdot \mathbf{P} + \pi_\mathbf{1} \cdot \gamma = (\mathbf{e_0} \cdot \mathbf{P} + \pi_\mathbf{0} \cdot \gamma) \cdot \mathbf{P} + (\pi_\mathbf{0} \cdot \mathbf{\Pi}) \cdot \gamma \\
&= \mathbf{e_0} \cdot \mathbf{P}^2 + \pi_\mathbf{0} \cdot (\gamma \cdot \mathbf{P} + \mathbf{\Pi} \cdot \gamma), \\
\mathbf{e_3} &= \mathbf{e_2} \cdot \mathbf{P} + \pi_\mathbf{2} \cdot \gamma = \left(\mathbf{e_0} \cdot \mathbf{P}^2 + \pi_\mathbf{0} \cdot (\gamma \cdot \mathbf{P} + \mathbf{\Pi} \cdot \gamma)\right) \cdot \mathbf{P} + \left(\pi_\mathbf{0} \cdot \mathbf{\Pi}^2\right) \cdot \gamma \\
&= \mathbf{e_0} \cdot \mathbf{P}^3 + \pi_\mathbf{0} \cdot \left(\gamma \cdot \mathbf{P}^2 + \mathbf{\Pi} \cdot \gamma \cdot \mathbf{P} + \mathbf{\Pi}^2 \cdot \gamma\right) \\
&= \mathbf{e_0} \cdot \mathbf{P}^3 + \pi_\mathbf{0} \cdot \sum_{i=0}^{2} \mathbf{\Pi}^i \cdot \gamma \cdot \mathbf{P}^{2-i}, \\
&\vdots \\
\mathbf{e_k} &= \mathbf{e_0} \cdot \mathbf{P}^k + \pi_\mathbf{0} \cdot \sum_{i=0}^{k-1} \mathbf{\Pi}^i \cdot \gamma \cdot \mathbf{P}^{k-1-i}.
\end{aligned}
$$

We obtained a compact expression for calculating an (exact) error vector at step k based on initial approximate probability distribution $\pi_\mathbf{0}$. The first term represents a presence of the aggregation error that is being transported using unaggregated transition matrix. The second term decomposes the overall propagation error: a probability distribution $\pi_\mathbf{0}$ is first being propagated via aggregated transition matrix for i steps and then it produces an error, which is then being transported using the concrete transition matrix. This formula cannot be applied directly with the state-space aggregation since we do not want to compute $\gamma \cdot \mathbf{P}^{k-1-i}$ terms, but perhaps it can be simplified or approximated using some sort of decomposition. We currently lack the knowledge in this area, so we will leave as it is.

5.3 Conclusions

In this work we provided a comparative analysis of approximative techniques for Markov chain analysis, namely, methods based on aggregation or truncation. We first focused on the design of an accurate and efficient aggregation method applicable to chains with an arbitrary structure of the state space. We started in discrete setting and redefined a notion of the state-space abstraction in order to arrive at precise bounds on the approximation error. We then used these results to design a new aggregating scheme that preserves all properties of a Markov chain and we showed that this preservation is necessary for integrating it with uniformisation method to enable analysis of continuous-time models. This integration was then carried out, and explicit bounds on the approximation error were derived. Finally, we introduced adaptivity to our aggregating scheme that allowed reducing the required number of computation steps.

A total of eight approximative methods for Markov chain analysis (5 existing and 3 new ones) were implemented in the probabilistic model checker PRISM and were also integrated with the model checking algorithms. Experiments confirm that newly derived bounds provide a several orders of magnitude precision improvement without degrading performance. We show that the resulting aggregating approach can provide a valid model approximation supplied by adequate approximation error estimates, in both discrete and continuous cases. Then, we perform a comparative analysis of aggregating and truncating techniques, illustrate how different methods handle various types of models and identify chains for which aggregating, or truncating, analysis is preferred. In particular, we prove that the designed

aggregation scheme is capable of outperforming existing methods, including FAU. Finally, we demonstrate a successful usage of approximative techniques for model checking Markov chains. Future work will include a further development of the error estimates, performance profiling and improvement upon existing algorithms, as well as effective combination of the approximative techniques with parameter synthesis or verification procedures.

Bibliography

[1] Abate, A.; Brim, L.; Češka, M.; et al.: Adaptive Aggregation of Markov Chains: Quantitative Analysis of Chemical Reaction Networks. In *Computer Aided Verification*. Cham: Springer International Publishing. 2015. ISBN 978-3-319-21690-4. pp. 195–213.

[2] Baier, C.; Hanh, E. M.; Haverkort, B. R.; et al.: Model checking for performability. *Mathematical Structures in Computer Science*. vol. 23, no. 4. 2013: page 751–795. doi:10.1017/S0960129512000254.

[3] Bertsekas, D.; Tsitsiklis, J.: *Introduction to Probability*. Athena Scientific books. Athena Scientific. 2002. ISBN 9781886529403.
Retrieved from: https://books.google.cz/books?id=bcHaAAAAMAAJ

[4] Bolch, G.; Greiner, S.; de Meer, H.; et al.: *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. New York, NY, USA: Wiley-Interscience. 1998. ISBN 0-471-19366-6.

[5] Cardelli, L.; Kwiatkowska, M.; Whitby, M.: *Chemical Reaction Network Designs for Asynchronous Logic Circuits*. Cham: Springer International Publishing. 2016. ISBN 978-3-319-43994-5. pp. 67–81. doi:10.1007/978-3-319-43994-5_5.
Retrieved from: http://dx.doi.org/10.1007/978-3-319-43994-5_5

[6] Dannenberg, F.; Hahn, E. M.; Kwiatkowska, M.: Computing Cumulative Rewards Using Fast Adaptive Uniformization. *ACM Trans. Model. Comput. Simul.*. vol. 25, no. 2. February 2015: pp. 9:1–9:23. ISSN 1049-3301. doi:10.1145/2688907.
Retrieved from: http://doi.acm.org/10.1145/2688907

[7] Didier, F.; Henzinger, T. A.; Mateescu, M.; et al.: Fast Adaptive Uniformization of the Chemical Master Equation. In *2009 International Workshop on High Performance Computational Systems Biology*. Oct 2009. pp. 118–127. doi:10.1109/HiBi.2009.23.

[8] Esmaeil Zadeh Soudjani, S.; Abate, A.: Precise Approximations of the Probability Distribution of a Markov Process in Time: An Application to Probabilistic Invariance. In *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2014. ISBN 978-3-642-54862-8. pp. 547–561.

[9] Ferm, L.; Lötstedt, P.: Adaptive solution of the master equation in low dimensions. *Applied Numerical Mathematics*. vol. 59, no. 1. 2009: pp. 187 – 204. ISSN 0168-9274. doi:https://doi.org/10.1016/j.apnum.2008.01.004.
Retrieved from:
http://www.sciencedirect.com/science/article/pii/S0168927408000263

[10] Fox, B. L.; Glynn, P. W.: Computing Poisson Probabilities. *Commun. ACM*. vol. 31, no. 4. April 1988: pp. 440–445. ISSN 0001-0782. doi:10.1145/42404.42409.
Retrieved from: http://doi.acm.org/10.1145/42404.42409

[11] Galil, Z.; Italiano, G. F.: Data Structures and Algorithms for Disjoint Set Union Problems. *ACM Comput. Surv.*. vol. 23, no. 3. September 1991: pp. 319–344. ISSN 0360-0300. doi:10.1145/116873.116878.
Retrieved from: http://doi.acm.org/10.1145/116873.116878

[12] Gillespie, D. T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*. vol. 22, no. 4. 1976: pp. 403 – 434. ISSN 0021-9991.
doi:https://doi.org/10.1016/0021-9991(76)90041-3.
Retrieved from:
http://www.sciencedirect.com/science/article/pii/0021999176900413

[13] Gillespie, D. T.: Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*. vol. 81, no. 25. 1977: pp. 2340–2361.
doi:10.1021/j100540a008.
Retrieved from: https://doi.org/10.1021/j100540a008

[14] Hey, J.; Nielsen, R.: Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics. *Proceedings of the National Academy of Sciences*. vol. 104, no. 8. 2007: pp. 2785–2790. ISSN 0027-8424.
doi:10.1073/pnas.0611164104.
Retrieved from: http://www.pnas.org/content/104/8/2785

[15] Hoops, S.; Sahle, S.; Gauges, R.; et al.: COPASI—a COmplex PAthway SImulator. *Bioinformatics*. vol. 22, no. 24. 2006: pp. 3067–3074.
doi:10.1093/bioinformatics/btl485.
Retrieved from: http://dx.doi.org/10.1093/bioinformatics/btl485

[16] Kierzek, A. M.; Zaim, J.; Zielenkiewicz, P.: The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression. *The Journal of biological chemistry*. vol. 276 11. 2001: pp. 8165–72.

[17] Kwiatkowska, M.; Norman, G.; Parker, D.: PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Computer Aided Verification*, *LNCS*, vol. 6806. Springer. 2011. pp. 585–591.

[18] Larsen, K. G.; Skou, A.: Bisimulation through probabilistic testing. *Information and Computation*. vol. 94, no. 1. 1991: pp. 1 – 28. ISSN 0890-5401.
doi:http://dx.doi.org/10.1016/0890-5401(91)90030-6.
Retrieved from:
http://www.sciencedirect.com/science/article/pii/0890540191900306

[19] Madsen, C.; Myers, C. J.; Roehner, N.; et al.: Utilizing stochastic model checking to analyze genetic circuits. In *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. May 2012. pp. 379–386.
doi:10.1109/CIBCB.2012.6217255.

[20] Malliaros, F. D.; Vazirgiannis, M.: Clustering and Community Detection in Directed Networks: A Survey. *CoRR*. vol. abs/1308.0971. 2013.

[21] van Moorsel, A. P. A.; Sanders, W. H.: Adaptive uniformization. *Communications in Statistics. Stochastic Models*. vol. 10, no. 3. 1994: pp. 619–647. doi:10.1080/15326349408807313.
Retrieved from: https://doi.org/10.1080/15326349408807313

[22] Schwertman, N. C.; Gilks, A. J.; Cameron, J.: A Simple Noncalculus Proof That the Median Minimizes the Sum of the Absolute Deviations. *The American Statistician*. vol. 44, no. 1. 1990: pp. 38–39. doi:10.1080/00031305.1990.10475690.
Retrieved from:
https://amstat.tandfonline.com/doi/abs/10.1080/00031305.1990.10475690

[23] Stewart, W.: *Introduction to the numerical solution of Markov chains*. Princeton, NJ: Princeton Univ. Press. 1994. ISBN 0691036993.
Retrieved from: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+152880593&sourceid=fbw_bibsonomy

[24] T Gillespie, D.: Stochastic Simulation of Chemical Kinetics. vol. 58. 02 2007: pp. 35–55.

[25] Češka, M.; Šafránek, D.; Dražan, S.; et al.: Robustness Analysis of Stochastic Biochemical Systems. *PLOS ONE*. vol. 9, no. 4. 04 2014: pp. 1–23. doi:10.1371/journal.pone.0094553.
Retrieved from: https://doi.org/10.1371/journal.pone.0094553