Efficient Simulation Techniques for Stochastic Model Checking

Daniël Reijsbergen

Efficient Simulation Techniques for Stochastic Model Checking

Daniël Reijsbergen

Graduation committee:

Chairman:	prof. dr. ir. A.J. Mouthaan
Promoter:	prof. dr. ir. B.R.H.M. Haverkort
Promoter:	prof. dr. R.J. Boucherie
Assistant promoter:	dr. ir. P.T. de Boer
Assistant promoter:	dr. ir. W.R.W. Scheinhardt

Members:

dr. G. Rubino	INRIA Rennes / IRISA
prof. dr. M.R.H. Mandjes	University of Amsterdam
dr. A.A.N. Ridder	Vrije Universiteit Amsterdam
prof. dr. J.C. van de Pol	University of Twente
dr. ir. E.A. van Doorn	University of Twente

CTIT

CTIT Ph.D.-thesis Series No. 13-281
Centre for Telematics and Information Technology
University of Twente
P.O. Box 217, NL – 7500 AE Enschede

ISSN 1381-3617
ISBN 978-90-365-3586-1

Printed by Gildeprint Drukkerijen
Cover fractal designed using Apophysis

This work is supported by the Netherlands Organisation for Scientific Research (NWO), project no. 612.064.812.

EFFICIENT SIMULATION TECHNIQUES FOR STOCHASTIC MODEL CHECKING

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
Prof. dr. H. Brinksma,
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 6 december 2013 om 14.45 uur

door

Daniël Petrus Reijsbergen

geboren op 10 september 1985
te 's Gravenhage, Nederland

Dit proefschrift is goedgekeurd door:
prof. dr. ir. B.R.H.M. Haverkort (promotor)
prof. dr. R.J. Boucherie (promotor)
dr. ir. P.T. de Boer (assistent promotor)
dr. ir. W.R.W. Scheinhardt (assistent promotor)

Abstract

In this thesis, we focus on methods for speeding-up computer simulations of stochastic models. We are motivated by real-world applications in which corporations formulate service requirements in terms of an upper bound on a probability of failure. If one wants to check whether a complex system model satisfies such a requirement, computer simulation is often the method of choice. We aim to aid engineers during the design phase, so a question of both practical and mathematical relevance is how the runtime of the simulation can be minimised.

We focus on two settings in which a speed-up can be achieved. First, when the probability of failure is low, as is typical in a highly reliable system, the time before the first failure is observed can be impractically large. Our research involves *importance sampling*; we simulate using a different probability measure under which failure is more likely, which typically decreases the variance of the estimator. In order to keep the estimator unbiased, we compensate for the resulting error using the Radon-Nikodym theorem. If done correctly, the gains can be huge. However, if the new probability measure is unsuited for the problem setting the negative consequences can be similarly profound (infinite variance is even possible). In our work, we have extended an importance sampling technique with good performance (i.e., proven to have bounded relative error) that was previously only applicable in restricted settings to a broad model class of stochastic (Markovian) Petri nets. We have also proposed methods to alleviate two well-known problems from the rare event simulation literature: the occurrence of so-called high-probability cycles and the applicability to large time horizons. For the first we use a method based on Dijkstra's algorithm, for the second we use renewal theory.

Second, it often occurs that the number of needed simulation runs is overestimated. As a solution, we use *sequential hypothesis testing*, which allows us to stop as soon as we can say whether the service requirement is satisfied. This area has seen a lot of recent interest from the model checking community, but some of the techniques used are not always perfectly understood. In our research we have compared the techniques implemented in the most popular model checking tools, identified several common pitfalls and proposed a method that we proved to not have this problem. In particular, we have proposed a new test for which we bounded the probability of an incorrect conclusion using martingale theory.

Samenvatting

In dit proefschrift richten we ons op methoden om computersimulaties van stochastische modellen te versnellen. Onze motivatie vinden we in praktijksituaties waarin bedrijven een serviceniveau garanderen in termen van een bovengrens op de kans dat hun dienst niet meer (volledig) kan worden geleverd. Om te controleren of een complex systeemmodel aan een dergelijke eis voldoet heeft computersimulatie als methode veelal de voorkeur. Aangezien we ingenieurs tijdens de ontwerpfase trachten te helpen is een zowel vanuit praktisch als wiskundig oogpunt interessante vraag hoe we de duur van de simulaties zoveel mogelijk kunnen verkorten.

We richten ons op twee situaties waarin een versnelling kan worden bereikt. Ten eerste geldt dat als de kans op systeemfalen klein is, wat men mag verwachten in een *highly reliable system*, het onpraktisch lang kan duren totdat de eerste faalgebeurtenis zich voordoet in de simulatie. In ons onderzoek passen we *importance sampling* toe; we simuleren onder een kansmaat waarin systeemfalen waarschijnlijker wordt gemaakt, wat in het algemeen de variantie van de schatter verkleint. Teneinde de schatter zuiver te houden compenseren we middels de Radon-Nikodym-stelling voor de resulterende afwijking. Als dit goed wordt uitgevoerd kan de winst enorm zijn. Als de nieuwe kansmaat echter ongeschikt is voor de probleemsituatie kunnen de negatieve consequenties even ingrijpend zijn (oneindige variantie is mogelijk). In dit proefschrift hebben we een goed presterende simulatietechniek (waarvoor bewezen is dat het de eigenschap van *bounded relative error* bezit) voor zeldzame gebeurtenissen ('*rare events*'), die voorheen slechts in beperkte gevallen toepasbaar was, uitgebreid naar de brede modelklasse van stochastische (Markovse) Petri-netten. We hebben tevens methoden voorgesteld om twee bekende problemen in de literatuur omtrent simulatie van *rare events* te verhelpen: het voorkomen van zogeheten *high-probability cycles* en de toepasbaarheid op modellen die lange tijdsduren bestrijken. Voor het eerste geval gebruikten we een methode gebaseerd op Dijkstra's algoritme, voor de tweede gebruikten we vernieuwingstheorie.

Ten tweede komt het vaak voor dat wordt overschat hoe vaak men de simulatie moet draaien. Als oplossing gebruiken we sequentiële hypothesetoetsing, wat ons in staat stelt op te houden zodra we kunnen zeggen of aan de garantie op het serviceniveau is voldaan. Dit onderzoeksgebied heeft recentelijk veel aandacht gekregen vanuit de model-checkinggemeenschap, maar niet alle technieken worden altijd even goed begrepen. In ons onderzoek hebben we een aantal technieken met elkaar vergeleken die zijn geïmplementeerd in de meest populaire *model checking tools*, enkele gemeenschappelijke valkuilen benoemd en een methode voorgesteld waarvan we bewezen hebben dat die dit probleem omzeilt.

Contents

1 Introduction and Preliminaries **1**
- 1.1 Model Checking . . . 3
  - 1.1.1 Non-Probabilistic Model Checking . . . 4
  - 1.1.2 Probabilistic Model Checking . . . 5
  - 1.1.3 High-Level Model Description Languages . . . 7
  - 1.1.4 Algorithms for Probabilistic Model Checking . . . 10
  - 1.1.5 Model Assumptions . . . 12
- 1.2 Monte Carlo Simulation . . . 12
  - 1.2.1 Estimating path and steady-state probabilities . . . 13
  - 1.2.2 Path Generation . . . 14
- 1.3 Principles of Importance Sampling . . . 14
  - 1.3.1 Intuitive Description . . . 15
  - 1.3.2 Basic Setup of Importance Sampling . . . 15
  - 1.3.3 Variance of Importance Sampling Estimators . . . 16
  - 1.3.4 Failure Biasing and Forcing . . . 17
  - 1.3.5 Zero Variance . . . 17
- 1.4 Contributions of this Thesis . . . 18

2 Sequential Hypothesis Testing **21**
- 2.1 General Framework . . . 23
  - 2.1.1 Assumptions . . . 23
  - 2.1.2 Statistical Framework . . . 24
- 2.2 Known Statistical Hypothesis Tests . . . 27
  - 2.2.1 Gauss Test . . . 27
  - 2.2.2 Sequential Probability Ratio Test (SPRT) . . . 30
  - 2.2.3 Chernoff Test . . . 32
  - 2.2.4 Bayesian SPRT . . . 33
- 2.3 The New Supergaussian Sequential Tests . . . 35
  - 2.3.1 The Desired Shape of the Non-Critical (NC) Areas . . . 35
  - 2.3.2 Azuma Test . . . 36
  - 2.3.3 Darling Test . . . 40
  - 2.3.4 Optimal Parameter Choice . . . 41
- 2.4 Results and Comparisons . . . 43
  - 2.4.1 Shape of the Non-Critical Areas (NC) . . . 44
  - 2.4.2 Simulation Results . . . 45
- 2.5 Steady-State Measures . . . 49

2.6 Conclusions and Discussion . . . 51
2.6.1 Hypothesis Testing for Importance Sampling . . . 52

3 Rarity Regimes in the Birth-Death Process 55
3.1 The Birth-Death Process . . . 56
3.1.1 Definition . . . 56
3.1.2 Performance Measures . . . 57
The (Transient) Unreliability . . . 57
The Conditional Unreliability . . . 59
The Unavailability . . . 60
3.1.3 Straight Paths and Cycles . . . 61
3.2 Rarity Regimes in the Birth-Death Process . . . 62
3.2.1 The Regimes $\lambda \downarrow 0$ and $\bar{\tau} \downarrow 0$ (Slow Component Failures) . . . 62
3.2.2 The Regime $\mu \to \infty$ (Fast Component Repairs) . . . 64
3.2.3 The Regime $n \to \infty$ (Many Spares) . . . 64
3.2.4 The Regime $\bar{\tau} \to \infty$ (Large Mission Time) . . . 65
3.3 The Regime $\bar{\tau} \to \infty$: Fast Simulation for Slow Paths . . . 65
3.3.1 Conditional Sojourn Times . . . 66
3.3.2 Simulation . . . 68
3.3.3 Simulation Results . . . 69
3.4 Conclusions . . . 71

4 Networks of Parallel Birth-Death Processes 73
4.1 Standard Dedicated Repair . . . 73
4.1.1 Model Description . . . 74
4.1.2 The Distributed Database System (DDS) . . . 74
Model Description . . . 75
Operation and Failure . . . 76
The Benchmark Case and Generalisations . . . 77
4.1.3 Approximating $\pi_{\psi^{\bar{\tau}}}$ using Straight Paths . . . 77
4.1.4 Simulation Results . . . 79
Experimental Setup . . . 79
Unavailability . . . 80
Unreliability . . . 82
4.2 General Busy Cycle Durations . . . 84
4.2.1 Non-Memoryless Busy Cycles . . . 85
4.2.2 Non-Rare Paths . . . 87
4.2.3 Simulation Results . . . 89
Single Component Type . . . 89
Multiple Component Types . . . 89
4.3 Shared Repair Facilities . . . 92
4.4 Conclusions and Discussion . . . 94
4.4.1 General Conclusions . . . 94
4.4.2 Complexity . . . 95
4.4.3 Generalisations and Future Work . . . 96

5 Dominant Paths in General Markov Chains 97
5.1 Model Setting . . . 99
5.2 Dijkstra-based Algorithm . . . 102
5.2.1 Forward Phase . . . 102
5.2.2 Backward Phase . . . 104
5.2.3 Properties of the Algorithm . . . 105
5.3 Bounded Relative Error . . . 109
5.4 Vanishing Relative Error . . . 112
5.5 Time-Bounded Probabilities . . . 114
5.6 Conclusion . . . 119

6 Variance Reduction for Free 121
6.1 The Two New Methods . . . 121
6.2 Performance of the Methods . . . 123
6.2.1 Comparing $\hat{\pi}_\psi$ to $\hat{\pi}_\psi^+$ . . . 123
6.2.2 Comparing $\hat{\pi}_\psi$ to $\hat{\pi}_\psi^{++}$ . . . 124
6.2.3 Comparing $\hat{\pi}_\psi^+$ to $\hat{\pi}_\psi^{++}$ . . . 125
6.3 Illustrative Example . . . 125
6.4 Numerical Results . . . 129
6.4.1 Illustrative Example . . . 130
6.4.2 Two Path Model . . . 131
6.5 Conclusions . . . 134

7 Dominant Paths in Stochastic Petri Nets 135
7.1 Context within the literature . . . 136
7.2 Model Setting . . . 136
7.2.1 Reliability Modelling Using Petri Nets . . . 137
7.2.2 Running Example . . . 137
7.2.3 Problem Setting . . . 137
7.2.4 Efficient Simulation . . . 138
7.3 Determining the Distance Function . . . 139
7.3.1 Main Loop . . . 140
7.3.2 Initialisation Phase (`initZoneGraph`) . . . 140
7.3.3 Divide Zones by Path Existence (`possibilitySplit`) . . . 142
7.3.4 Divide Zones by Costs (`costSplit`) . . . 144
7.3.5 Update the step list (`update` and `alterS`) . . . 146
7.4 Empirical Results . . . 146
7.4.1 Case Description . . . 147
7.4.2 Results of the Distance Finding Algorithm . . . 147
7.4.3 Simulation Results . . . 147
7.5 Proof of Correctness . . . 149
7.6 Conclusions and Discussion . . . 153
7.6.1 Termination . . . 154
7.6.2 Integrality . . . 154

7.6.3 Importance Sampling Efficiency 155

8 Conclusions **157**
8.1 Contributions 157
8.2 Future Work 158
8.2.1 Hypothesis Testing 159
8.2.2 Automated Rare Event Simulation 160
8.2.3 Combining Hypothesis Testing and Rare Event Simulation . . 161

Bibliography **163**

List of Acronyms **171**

About the Author **173**

Acknowledgements **175**

CHAPTER 1

Introduction and Preliminaries

Society, and even human life increasingly depend on the faultless operation of technology-based products and services. Consequently, faults in software or hardware systems incur great costs. In the best case, these faults cause annoyances and, as a result, the producer or service provider loses money. In the worst case, people's safety is at stake [10]. Formal verification techniques for the analysis of system performance are becoming indispensable as a result of these developments. Unfortunately, perfect guarantees about the reliability of real-world systems are often impossible to give, so companies typically need to formulate requirements in terms of probabilities of failure. If companies can detect that their requirements are not met before their product (or service) reaches the final stages of production, this may allow them to make the necessary changes. However, being able to detect faults before actual deployment of the product requires an adequate model of the underlying system. Furthermore, it must be possible to compute or estimate the relevant probabilities in the model. This is usually done using a computer *tool*. As engineers need the tool during the design phase, it is vital that the runtime of the tool's algorithms is as small as possible. The focus of this thesis is to develop methods that reduce the runtime and which can be mathematically proven not to produce inaccurate results.

The use of verification techniques based on system models is called *model-based verification*. The most basic modelling formalism used for model-based verification techniques is that of *transition systems*: systems that consist of states and transitions that determine changes of the state. A specific realisation of the behaviour of the system model is then a sequence of state changes that we will call a *path*. Requirements on the system behaviour are specified, through a *formal language*, as requirements on paths. For *non-probabilistic* systems we use requirements given in terms of the existence of paths satisfying a certain requirement (e.g., there does not exist a path in which we reach deadlock before termination). In *probabilistic* systems, we are instead interested in probabilities of paths satisfying a certain requirement (e.g., the probability of seeing a path in which we reach deadlock before termination is less than 5%).

Model checking [10, 25] is (among others such as equivalence checking) a very popular model-based verification technique. A model checking algorithm exhaustively examines paths until a conclusion can be reached about whether the requirement is satisfied or not. For realistic models, the minimum number of paths that

need to be considered can be huge. As the number of (reachable) states in the system model increases the number of paths follows suit, possibly up to the point where numerical model checking algorithms become computationally infeasible. This is called the *state (space) explosion problem* [23]. One remedy is to use discrete-event simulation to generate a *random selection* of paths; based on this sample a *statistically* motivated statement can be made about the validity of the system requirement. This methodology is called *statistical model checking* [106, 115].

The main goal of this thesis is to achieve speed-ups for statistical model checking techniques. Focus will be on the investigation of properties that compare the probability that some *event* occurs to some boundary value. We focus on two main events. The first is the event that the system reaches some rare goal state before it reaches some more typical termination state. The second is the event that the system reaches some rare goal state before it reaches some more typical termination state *and* before some fixed time bound has been crossed. Additionally, in some parts of the thesis we will consider properties involving the long-term fraction of time that we reside in the set of goal states. We will formally specify all of these properties in the remainder of this chapter.

The speed-ups that we seek to achieve for statistical model checking fall into two main categories. First, it often occurs that the number of needed sample paths is chosen too large. As a solution, we use *sequential hypothesis testing* [69], which allows us to stop as soon as we can say whether the service requirement is satisfied. This area has seen a lot of recent interest from the model checking community and we aim to aid user understanding of the methods. Second, when the probability of failure is low, as is typical in a highly reliable system, we need to draw an impractically large number of samples before the first failure is observed. Our research involves *importance sampling* [50]; we simulate using a different probability measure under which failure is more likely, which typically speeds up the verification process. If done correctly, the gains can be huge. However, if the new probability measure is unsuited for the problem setting the negative consequences can be similarly profound. We aim to develop importance sampling techniques that have provably good performance for broad classes of modelling formalisms.

This thesis contains the following results for the two categories of speed-ups. We compare the hypothesis testing techniques implemented in the most popular model checking tools, identify several common potential pitfalls and propose a method that we proved to avoid these pitfalls. In particular, we propose a new test for which we bound the probability of an incorrect conclusion using martingale theory. Furthermore, we propose concrete rare event simulation techniques with provably good performance for several broad model classes. We also propose a novel technique that exploits information gained while constructing the new probability measure to achieve further speed-ups. A more detailed summary of contributions is given at the end of this chapter

In the rest of Chapter 1, we fix notation and discuss the position of our work in the broader context of the model checking and rare event simulation literature. We will begin in Section 1.1 by explaining model checking. We explain the formalisms

and languages that we will use in full generality, even though this thesis addresses only a subset of problems expressible in this manner. We maintain a very general scope for motivational purposes and to explain the position of our work in the context of the broader literature. We then explain standard simulation in Section 1.2 and rare event simulation in Section 1.3. In Section 1.4 we outline the remaining chapters of the thesis and highlight the contributions.

1.1 Model Checking

The basic model checking problem is the following: given a system model in the form of a transition system, does a given state x in the model satisfy some formally defined property Φ? If x does satisfy Φ, we write $x \vDash \Phi$, otherwise we write $x \nvDash \Phi$. Although many problems from applied mathematics can be cast into this framework, the defining characteristic of model checking algorithms is their generality. The aim is to define a model class and a *property specification language* that are as broad as possible such that all expressible problems can be solved by the same model checking algorithm.

Seminal in the field of model checking was the work of E. Clarke and E. Emerson on the so-called Computation Tree Logic (CTL) for non-probabilistic (temporal) models [22], for which they, together with J. Sifakis, received the 2007 Turing Award (the '*Nobel Prize*' of computer science). Their work was later extended with probabilities, resulting in the languages Probabilistic CTL (PCTL) [48] and Continuous Stochastic Logic (CSL) [4, 7].

In Section 1.1.1 we present CTL. Although non-probabilistic models will not appear later in the thesis, we have three reasons to do this: it provides motivational background, it makes the position of this thesis in the wider model checking literature more clear and the concepts from the non-probabilistic setting translate nicely to the probabilistic setting. In Section 1.1.2, we then discuss the extension to probabilistic systems; we discuss the type of probabilistic models that we are interested in (Markov models), and discuss PCTL and CSL. In Section 1.1.3, we describe the high-level language of stochastic Petri nets that we use to describe large (Markov) models in a concise way. In Section 1.1.4, we discuss the most common algorithms for probabilistic model checking and their strengths and weaknesses. In particular, we discuss simulation-based algorithms as they will be the focus of the rest of this thesis. In Section 1.1.5 we discuss several assumptions about the model class, and how they affect notation.

As a general note: although this section contains a discussion of model checking techniques in their full generality, the contents of the remainder of this thesis are largely described in terms of Markov chains. Hence, it may be preferable for some readers to avoid studying every part of this section in full detail (this is particularly true for Sections 1.1.1 and 1.1.2 and instead focus on the introduction of notation.

1.1.1 Non-Probabilistic Model Checking

For a computerised model checking tool to be able to validate whether a state in a system model satisfy a property, it is crucial that the property is unambiguously defined. CTL is a formal language with which to unambiguously express properties in non-probabilistic systems. It can be used to assert about a state x whether certain behaviour is possible or impossible starting from x. Examples of expressions in CTL include saying about state x that from x some desirable behaviour will always occur ('*liveness*') or that from x some undesirable behaviour will never occur ('*safety*'). In this section, we begin by formally introducing the model class before moving on to the property language CTL. Because we aim to keep notation as uniform as possible throughout the thesis, the way we introduce CTL is different from what is usually found in the model checking literature. Particularly, properties of interest will be divided into state and path properties[1], as we discuss later on.

The basic model for non-probabilistic model checking is a *Labelled Transition System (LTS)* [10]. Let A be a set of *atomic propositions*, fundamental properties of states which are assigned by the modeller to the states and which correspond to real-world interpretations of the state. E.g., A could be {sleeping, idle, busy} for a model of a signal processor or {empty, full} for a queueing model. Also, let 2^A be the power set of A. Then an LTS is a tuple ($\mathcal{X}$,T,$\mathcal{L}$) where $\mathcal{X}$ is a countable set of states called the *state space*, the matrix $T = (t_{xz})_{x,z\in\mathcal{X}}$ which is given by

$$t_{xz} = \begin{cases} 1 & \text{if there is a transition from } x \text{ to } z, \text{and} \\ 0 & \text{otherwise,} \end{cases}$$

gives the transition relation in the system, and $\mathcal{L} : \mathcal{X} \to 2^A$ is the labelling function which assigns to each state a set of atomic propositions. A path (of infinite length) ω is then a sequence $(x_0^\omega, x_1^\omega, \ldots)$ of states in $\mathcal{X}$.[2] Let Ω be the set of possible (infinite) paths, i.e., paths for which

$$\prod_{i=1}^{\infty} t_{x_{i-1}^\omega x_i^\omega} = 1.$$

Interesting behaviour in the system is then specified in terms of properties of the states and paths. These properties are given the form of state and path *formulae*. The state and path formulae that we use to define CTL will be given in the form of a context-free grammar [19]: essentially, this means that for both types of formulae we will list all possible expressions, where some expressions may recursively be defined by other expressions of the same form (e.g., an expression for Φ may contain Φ', which follows the same rules). Let a path formula ϕ in the non-probabilistic setting be given by

$$\phi := \mathsf{X}\,\Phi \mid \Phi\,\mathsf{U}\,\Phi',$$

[1] Path properties are formally the focus of the language LTL [85], but since we do not consider refined path properties, such as $\Diamond\Box\Phi$ that can occur in LTL, we restrict ourselves to the path properties needed to define CTL and its probabilistic extensions.

[2] We use the notation $(x_0^\omega, x_1^\omega, \ldots)$ instead of $(\omega_0, \omega_1, \ldots)$ for two reasons. First, we use $(\omega_1, \ldots, \omega_N)$ to denote a sample consisting of N sample paths. Second, we will later add transition times to the paths at which point we need to emphasise the states in the path anyway.

where Φ and Φ' are CTL-state formulae, which will be discussed below. The semantics for path formulae are as follows for each path $\omega \in \mathcal{X}$:

- $\omega \vDash \mathsf{X}\,\Phi$ iff $x_1^\omega \vDash \Phi$,
- $\omega \vDash \Phi\,\mathsf{U}\,\Phi'$ iff $\exists k \in \mathbb{N}$ s.t. $\forall j < k : x_j^\omega \vDash \Phi$ and $x_k^\omega \vDash \Phi'$.

We can define CTL using these path formulae. Let a CTL-state formula Φ be given by

$$\Phi := true \mid a \mid \neg\Phi' \mid \Phi' \vee \Phi'' \mid \mathsf{E}\,\phi \mid \mathsf{A}\,\phi,$$

where Φ' and Φ'' use the same grammar as Φ. The semantics for state formulae are as follows, for each state $x \in \mathcal{X}$ [10]:

- $x \vDash true$ for all $x \in \mathcal{X}$,
- $x \vDash a$ iff $a \in \mathcal{L}(x)$,
- $x \vDash \neg\Phi'$ iff $x \nvDash \Phi'$,
- $x \vDash \Phi' \vee \Phi''$ iff $x \vDash \Phi'$ or $x \vDash \Phi''$,
- $x \vDash \mathsf{E}\,\phi$ iff $\exists\omega$ s.t. $x_0^\omega = x$ and $\omega \vDash \phi$,
- $x \vDash \mathsf{A}\,\phi$ iff $\forall\omega$ s.t. $x_0^\omega = x$ it holds that $\omega \vDash \phi$.

Additionally, we use the following shorthand notation: $\mathsf{E}\,(\Diamond\Phi)$ for $\mathsf{E}\,(true\,\mathsf{U}\,\Phi)$ (there exists a path in which we *eventually* reach Φ) and $\mathsf{A}\,(\Diamond\Phi)$ for $\mathsf{A}\,(true\,\mathsf{U}\,\Phi)$ (in *all* paths we eventually reach Φ). Furthermore, we write $\mathsf{E}\,(\Box\Phi)$ for $\neg\,\mathsf{A}\,(\Diamond\neg\Phi)$ (there exists a path on which *globally* Φ holds) — the expression $\mathsf{A}\,(\Box\Phi)$ is defined analogously. This way, we can quickly express interesting system properties such as $\mathsf{A}\,(\Box\neg\text{deadlock})$ (i.e., it always holds that we are not in a deadlock) and $\mathsf{A}\,(\text{idle}\,\mathsf{U}\,\text{busy})$ (i.e., it always holds that at some point the system will become busy, and before that it was always idle). We can also derive common logical operators such as $\wedge$ and $\Rightarrow$ from $\neg$ and $\vee$.

Given a CTL-state formula Φ one uses graph-based algorithms based on fixed point computations to determine whether states satisfy Φ. A fundamental property of the algorithms used for CTL-model checking is that once they terminate, we know *for all states in* $\mathcal{X}$ whether they satisfy Φ. A result is that we can use these algorithms to verify so-called *nested* CTL-formulae such as $\mathsf{E}\,(\Diamond\,\mathsf{A}\,(\Box\Phi))$ (i.e., there exists a path such that we eventually reach a state in which it holds that all paths from that state globally satisfy Φ).

1.1.2 Probabilistic Model Checking

CTL model checking allows us to formulate hard guarantees about system performance such as "*the system will* never *deadlock*", but in practice such certainty may be unattainable. Instead of guaranteeing that there does not exist a path in which some undesirable event happens, we want to guarantee that the *probability* that such a path occurs is small. Hence, we need a notion of probabilities of paths. The widely used stochastic formalism that we consider in this thesis is that of the *Markov chain*.

In analogy with our definition of an LTS, let a (labelled) *Discrete-Time Markov Chain (DTMC)* [49] be a tuple $(\mathcal{X}, P, \mathcal{L})$ where $\mathcal{X}$ and $\mathcal{L}$ have the same interpretation as for the LTS, but where $P = (p_{xz})_{x,z\in\mathcal{X}}$ is a stochastic matrix, i.e.,

$$0 \leq p_{xz} \leq 1 \ \forall x, z \in \mathcal{X} \ \text{ and } \ \sum_{z\in\mathcal{X}} p_{xz} = 1 \ \forall x \in \mathcal{X}.$$

The evolution of the state of the DTMC in an execution path ω is then a stochastic process $(X_i^\omega)_{i\in\mathbb{N}}$ governed by the probability measure $\mathbb{P}$ defined as follows. If, during an execution of the system, we have so far observed a path $(x_0^\omega, x_1^\omega, \ldots, x_i^\omega)$, then

$$\mathbb{P}\left(X_{i+1}^\omega = z \,\middle|\, X_i^\omega = x_i^\omega, \ldots, X_0^\omega = x_0^\omega\right) = \mathbb{P}\left(X_{i+1}^\omega = z \,\middle|\, X_i^\omega = x_i^\omega\right) = p_{x_i^\omega z}$$

The probability of observing a certain finite path $\omega = (x_0^\omega, x_1^\omega, \ldots, x_{|\omega|}^\omega)$ is then given by

$$\mathbb{P}(\omega) = \prod_{i=1}^{|\omega|} p_{x_{i-1}^\omega x_i^\omega}.$$

One easily shows by induction that $\mathbb{P}$ defines a valid probability measure given a starting state x_0^ω — we write $\mathbb{P}_{x_0^\omega}$ if the starting state is not clear from the context and otherwise $\mathbb{P}$ for brevity. The probability of a path of infinite length follows by taking the limit $|\omega| \to \infty$ (this probability is non-zero if a state is reached for which all subsequent transitions occur with probability 1. A common example is an *absorbing* state: a state x such that $p_{xx} = 1$). Let an *event* be a set of paths. The probability of an event is then simply the sum of the probabilities of its elements. Since the satisfaction of a CTL-path formula is an event, we can now reason about probabilities of satisfying path formulae. This gives rise to the property specification language for DTMCs called *probabilistic CTL* (PCTL). For PCTL-state formulae, the existential and universal quantifiers E and A are replaced by the probabilistic operator $\mathcal{P}_{\bowtie p}(\phi)$, which holds for a state $x \in \mathcal{X}$ iff $\mathbb{P}_x(\{\omega : \omega \vDash \phi\}) \bowtie p$ with $\bowtie \in \{\leq, \geq\}$. Also, a new path formula is added, namely the *bounded until* formula $\Phi \,\mathsf{U}^{\leq m} \Phi'$, which holds for a path ω iff $\exists k \leq m$ s.t. $\forall j < k : \omega_j \vDash \Phi$ and $\omega_k \vDash \Phi'$. This is summarised in Table 1.1.

A second, alternative notion of probabilistic behaviour in an LTS is that of the (labelled) *Continuous-Time Markov Chain (CTMC)*. A CTMC is a DTMC augmented with continuous (specifically: exponentially distributed) transition times. Hence, a path $\boldsymbol{\omega}$ in a CTMC is *timed*, i.e., it is a sequence of states and of (increasing) time points at which state changes occur, written as $\boldsymbol{\omega} = (x_0^{\boldsymbol{\omega}}, \tau_1^{\boldsymbol{\omega}}, x_1^{\boldsymbol{\omega}}, \ldots, x_{|\boldsymbol{\omega}|}^{\boldsymbol{\omega}}, \tau_{|\boldsymbol{\omega}|}^{\boldsymbol{\omega}})$, where for all $i \in \{0, \ldots, |\boldsymbol{\omega}|\}$ it holds that $x_i^{\boldsymbol{\omega}} \in \mathcal{X}$ and $\tau_i^{\boldsymbol{\omega}} \in \mathbb{R}^+$ (we fix $\tau_0^{\boldsymbol{\omega}} = 0$ for all $\boldsymbol{\omega}$). Specifically, $\tau_i^{\boldsymbol{\omega}}$ is a realisation of $\mathcal{T}_i^{\boldsymbol{\omega}}$, where $\mathcal{T}_i^{\boldsymbol{\omega}}$ is the random variable defined to be the time point in execution path $\boldsymbol{\omega}$ at which the system moves from state $x_{i-1}^{\boldsymbol{\omega}}$ to state $x_i^{\boldsymbol{\omega}}$. Let $\boldsymbol{\Omega}$ be the set of all timed paths.

The CTMC itself is a tuple $(\mathcal{X}, P, \eta, \mathcal{L})$, where $\eta : \mathcal{X} \to (0, \infty)$ such that, for $x \in \mathcal{X}$, $\eta(x)$ (also written as η_x) is the *exit rate* of state x. The time $\mathcal{T}_{i+1} - \tau_i$ that

we spend in state x_i is then exponentially distributed with rate η_{x_i}, and the *rate* of jumping from state x_i to x_{i+1} is given by $\eta(x_i) \cdot p_{x_i x_{i+1}}$. Probabilities on timed paths are then given by the probability measure $\mathbb{P}$ for CTMCs which is uniquely defined by the relation in which, if we have so far observed a path $(x_0^{\boldsymbol{\omega}}, \tau_1^{\boldsymbol{\omega}}, x_1^{\boldsymbol{\omega}}, \ldots, x_i^{\boldsymbol{\omega}}, \tau_i^{\boldsymbol{\omega}})$,

$$\mathbb{P}\left(X_{i+1}^{\boldsymbol{\omega}} = z,\, \mathcal{T}_{i+1}^{\boldsymbol{\omega}} - \mathcal{T}_i^{\boldsymbol{\omega}} < t \,\middle|\, (X_0^{\boldsymbol{\omega}}, \mathcal{T}_1^{\boldsymbol{\omega}}, \ldots, X_i^{\boldsymbol{\omega}}, \mathcal{T}_i^{\boldsymbol{\omega}}) = (x_0^{\boldsymbol{\omega}}, \tau_1^{\boldsymbol{\omega}}, \ldots, x_i^{\boldsymbol{\omega}}, \tau_i^{\boldsymbol{\omega}})\right) = \mathbb{P}\left(X_{i+1}^{\boldsymbol{\omega}} = z,\, \mathcal{T}_{i+1}^{\boldsymbol{\omega}} - \tau_i^{\boldsymbol{\omega}} < t \,\middle|\, X_i^{\boldsymbol{\omega}} = x_i^{\boldsymbol{\omega}}\right) = p_{x_i^{\boldsymbol{\omega}} z}\left(1 - e^{-\eta(x_i^{\boldsymbol{\omega}})t}\right).$$

An example of a CTMC can be found in Figure 3.1.

The property specification language for CTMC is the *Continuous Stochastic Logic* (CSL). It is similar to PCTL, except that it also has a bounded next X^I alongside the bounded until, and the notion of the time bounds is now continuous instead of discrete. More detail can be found in Table 1.1. Also, for CSL the *steady-state operator* $\mathcal{S}_{\bowtie p}(\Phi)$ was introduced, which is satisfied in state x iff $\lim_{k\to\infty} \mathbb{P}_x(\{\omega : \omega_k \vDash \Phi\})$ exists and is $\bowtie\ p$. If the underlying Markov chain is strongly connected, then this property is either satisfied in all states or in none, but if $\mathcal{X}$ consists of more than one disjoint strongly connected component then its satisfaction may be state-dependent.

Throughout this thesis, we will use the shorthand notation $\mathbb{P}_x(\phi) = \mathbb{P}_x(\{\omega : \omega \vDash \phi\})$ where ϕ can be a PCTL- or a CSL-path formula.

1.1.3 High-Level Model Description Languages

While the core modelling formalisms used for PCTL and CSL model checking those of Markov chains, the Markov chains themselves are often not specified explicitly at the state level. Instead, one uses a higher (architectural) level from which the underlying Markov chain can be generated automatically using well-known state-space exploration algorithms. Typically, high-level descriptions are closer to the view of the system designer and they allow models consisting of a vast number of states to be specified with relative ease. A downside is that the size of the state space is sometimes larger than necessary (i.e., they may contain clusters of states that may be *lumped* [32]). However, these languages are very well suited for techniques that do not require explicit knowledge of the state space, such as the simulation-based techniques to be discussed in Section 1.1.4.

A common high-level description language is the *Architecture Analysis & Design Language (AADL)* [38]. To make dependability analysis possible, AADL-based models can be transformed [101] into more basic high-level models such as a *Stochastic Petri Net (SPN)*. SPNs will be the only high level models that we consider in this thesis (mainly in Chapter 7). SPNs are the probabilistic extension of the non-probabilistic formalism of the Petri net, the same way PCTL and CSL compare to CTL. A formal definition is given below.

In the following, we will describe Multi-Guarded Petri Nets as in [58], although we extend the net with marking-dependent firing rates for the transitions. We only discuss the concepts of SPNs that we need for the rest of the thesis; for a

state properties: $x \vDash \Phi$ iff			
model	non-prob. LTSs	DTMCs	CTMCs
language	CTL	PCTL	CSL
$\Phi = true$	$x \in \mathcal{X}$		
$\Phi = a$	$a \in \mathcal{L}(x)$		
$\Phi = \neg\Phi'$	$x \nvDash \Phi'$		
$\Phi = \Phi' \vee \Phi''$	$x \vDash \Phi'$ or $x \vDash \Phi''$		
$\Phi = \mathsf{E}\,\phi$	$\exists\omega$ s.t. $\omega_0 = x$ and $\omega \vDash \phi$	✗[1]	
$\Phi = \mathsf{A}\,\phi$	$\forall(\omega$ s.t. $\omega_0 = x) : \omega \vDash \phi$	✗[1]	
$\Phi = \mathcal{P}_{\bowtie p}(\phi)$	✗	$\mathbb{P}_x(\{\omega : \omega \vDash \phi\}) \bowtie p$	
$\Phi = \mathcal{S}_{\bowtie p}(\Phi)$	✗	$\lim_{k\to\infty} \mathbb{P}_x(\{\omega : x^\omega_k \vDash \Phi\}) \bowtie p$ [2]	

path properties: $\omega \in \Omega \vDash \phi$ iff			
language	CTL	PCTL	CSL
$\phi =$ X	$x^\omega_1 \vDash \Phi$		✗
$\phi =$ X^I	✗		$x^\omega_1 \vDash \Phi$ and $\tau^{\boldsymbol{\omega}}_{\lvert\boldsymbol{\omega}\rvert} \in I$
$\phi =$ $\Phi\,\mathsf{U}\,\Phi'$	$\exists k \in \mathbb{N}$ s.t. $\forall j < k : x^\omega_j \vDash \Phi$ and $x^\omega_k \vDash \Phi'$		
$\phi =$ $\Phi\,\mathsf{U}^I\,\Phi'$	✗	$\exists k \in I$ s.t. $\forall j < k :$ $x^\omega_j \vDash \Phi$ and $x^\omega_k \vDash \Phi'$	$\exists k \in \mathbb{N}$ s.t. $\forall j < k :$ $x^\omega_j \vDash \Phi$ and $x^\omega_k \vDash \Phi'$ and $\tau^{\boldsymbol{\omega}}_{\lvert\boldsymbol{\omega}\rvert} \in I$

Table 1.1: Summary of the languages CTL, PCTL and CSL.
[1] Some CTL-formulae do not have an equivalent in PCTL. E.g., $\mathsf{A}\,(\Diamond\Phi)$ has no equivalent: the only obvious candidate, $\mathcal{P}_{\geq 1}(\Diamond\Phi)$, is not equivalent as there may exist infinite paths that do not satisfy ϕ, yet which occur with probability 0.
[2] Originally, $\mathcal{S}_{\bowtie p}(\Phi)$ was not in PCTL [48]. However, its meaning also makes sense in DTMCs and model checking tools such as PRISM allow this expression in PCTL. In fact, for CTMCs, for which the operator was originally proposed, the verification of this formula is often done by reducing the CTMC to its uniformised DTMC [9].

more extensive treatment of basic Petri nets see, e.g., [79]. We define an SPN to be $(P, T, Pre, Post, G, \vec{\lambda})$, where

- $P = \{1, 2, \ldots, |P|\}$ denotes the set of *places*,
- $T = \{t_1, \ldots, t_{|T|}\}$ denotes the set of *transitions*,
- $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre-* and *post- incidence functions* (as we will explain later).[3]
- G denotes the set of *guards*.

[3] We use $\mathbb{N} = \{0, 1, 2, \ldots\}$.

- $\vec{\lambda} = (\lambda_1, \ldots, \lambda_{|T|})$ denotes the *rate functions*, where $\lambda_i : \mathbb{N}^{|P|} \to \mathbb{R}^+$ represents the rate at which transition t_i fires as a function of the current marking (which is in $\mathbb{N}^{|P|}$).

Let $X_i(n)$ be the number of tokens in place i after the nth time a transition is fired, $n \in \mathbb{N}$. Let $\vec{X}(n) = (X_1(n), \ldots, X_{|P|}(n))$ be the *marking* (or *state*) of the SPN at time n. The state space $\mathcal{X} = \mathbb{N}^{|P|}$ is now the set of all possible markings; since $\mathcal{X}$ is now a set of vectors, we will from now on write states as $\vec{x} = (x_1, \ldots, x_{|P|})$ instead of x. We let transition t_i have exponential rate $\lambda_i(\vec{x})$ with $\vec{x} \in \mathcal{X}$. The rate $\lambda_i(\vec{x}(n))$ determines the relative likelihood of the transition to *fire* at step n (if it is enabled).

When transition t fires, the marking changes as follows: $Pre(p,t)$ tokens are removed from place p while $Post(p',t)$ tokens are added to place p'. A transition cannot fire if this would result in a negative number of tokens in a place, nor can it fire when one of its guards is not enabled (as discussed below). The guards can be described in terms of *constraints*, a concept that we will use extensively in Chapter 7. A constraint $c = (\vec{\alpha}, \beta, \bowtie)$ is an element of $\mathbb{Z}^{|P|} \times \mathbb{Z} \times \{\leq, \geq\}$,[4] and we say that marking $\vec{x}$ satisfies constraint c if $\vec{\alpha}^{\mathrm{T}}\vec{x} \bowtie \beta$. A *guard* g is then a 4-tuple $(p, t, \beta, \bowtie)$ that imposes upon a transition t the necessary condition that it can only fire in $\vec{x}(n)$ if the number of tokens in place p satisfies the inequality $x_p(n) \bowtie \beta$. Letting

$$\mathbf{1}_i(\vec{x}(n)) = \begin{cases} 1 & \text{if } \forall (p, t_i, \beta, \bowtie) \in G : x_p(n) \bowtie \beta, \\ 0 & \text{otherwise,} \end{cases} \tag{1.1}$$

we say that transition t_i is enabled at time n if $\mathbf{1}_i(\vec{x}(n)) = 1$.[5] If there are no guards $g \in G$ such that $g = (\cdot, t, \cdot, \cdot)$ then the transition t is *always* enabled as long as firing t does not result in a negative number of tokens in some place. Let the total *incidence vector* $\vec{u}_i = (u_{i1}, \ldots, u_{i|P|})$ of transition t_i be the vector that describes the effect of firing t_i on the marking. It is defined by $u_{ij} = Post(j,i) - Pre(j,i)$, hence the names pre- and post-incidence functions. Then the marking process $\vec{X}(n)$ is a DTMC that is uniquely characterised by the probability measure

$$\begin{aligned} \mathbb{P}(\vec{x}(n) \to \vec{x}(n+1)) &= \mathbb{P}\left(\vec{X}(n+1) = \vec{x}(n+1) \,\middle|\, \vec{X}(n) = \vec{x}(n)\right) \\ &= \frac{\sum_{i \in \mathcal{I}} \lambda_i(\vec{x}(n)) \mathbf{1}_i(\vec{x}(n))}{\sum_{j=1}^{|T|} \lambda_j(\vec{x}(n)) \mathbf{1}_j(\vec{x}(n))}, \end{aligned} \tag{1.2}$$

where $\mathcal{I} = \{i \in \mathbb{N} : t_i \in T,\ \vec{x}(n+1) = \vec{x}(n) + \vec{u}_i\}$. We will often use the short-hand notation $p_{\vec{x}}(j) \triangleq \mathbb{P}(\vec{x} \to \vec{x} + \vec{u}_j)$. An example of an SPN can be found in Figure 7.1.

To construct a CTMC using an SPN, we use $\mathbb{P}$ of (1.2) for the transition probabilities and the denominator in (1.2) as the exit rate in state $\vec{x}(n)$, i.e.,

$$\eta(\vec{x}(n)) = \sum_{j=1}^{|T|} \lambda_j(\vec{x}(n)) \mathbf{1}_j(\vec{x}(n)).$$

[4] We restrict ourselves to integer-valued constraints because this is necessary for our analysis in Chapter 7.

[5] Note that guards are only allowed to depend on the number of tokens in a single place, e.g., a guard of the form $x_1 + x_2 \leq 3$ is not allowed.

1.1.4 Algorithms for Probabilistic Model Checking

The algorithms for probabilistic model checking can be divided into two categories: *numerical* and *statistical* techniques. The numerical algorithms for $\mathcal{S}_{\bowtie p}(\Phi)$ (about the steady-state/limiting probability of being in Φ-states) and $\mathcal{P}_{\bowtie p}(\Phi \,\mathsf{U}\, \Phi')$ (about the probability of visiting a Φ'-state before a $\neg\Phi$-state) can be expressed using a matrix equation involving P or another matrix of a similar size. In model checking tools such as PRISM, this matrix equation is then solved using iterative techniques such as the Jacobi and Gauss-Seidel methods [68]. For $\mathcal{P}_{\bowtie p}(\Phi \,\mathsf{U}^{I}\, \Phi')$ (about the probability of visiting a Φ'-state before a $\neg\Phi$-state within the time interval I), the probabilities of interest are computed using simple matrix multiplications for DTMCs or uniformisation [9] for CTMCs.[6] Numerical algorithms have several well-known drawbacks.

a. The state space explosion problem: the complexity of all previously mentioned algorithms depends on the size[7] of the probability transition matrix P. When the state space is large or even infinite, a naive implementation of these algorithms may be computationally infeasible. There is a vast literature on ways to mitigate this problem; a lot of focus has been on state space reduction techniques such as partial order reduction [24] or confluence [13], or symbolic representations of the state space [77]. Furthermore, in some cases when the state space is infinite [95] more refined techniques exist which may lead to a solvable problem. However, many of these more refined techniques require forms of structure in the model that are not always present.

b. Stiffness in CTMC model checking: when there are states in the CTMC that have a very high exit rate, the uniformisation rate that is used to turn the CTMC into a DTMC is also large. This means that the other states will be given the same exit rate combined with a very large self-loop probability. This has an adverse effect on the performance of the techniques used to analyse the resulting DTMC [15].

c. The inability to generalise these techniques to adjacent settings. For example, when rewards are added to the model (the model is then a Markov Reward Model and the associated property specification language is CSRL [8]), more refined techniques are required that suffer even worse when the size of the state space increases — for models with over 5000 states [26] these techniques are currently out of scope. Also, when intertransition times are allowed to have probability distributions other than the exponential (which is done in, e.g., some models of probabilistic timed automata [29]), techniques designed for Markov chains fail to be applicable altogether.

Still, when a numerical model checking algorithm terminates one knows with certainty (or in some cases up to a bounded error) that the property of interest is sat-

[6] Model checking the bounded and unbounded X-operator of PCTL and CSL is largely trivial and will not be discussed in this thesis.

[7] More specifically: on the number of non-zeroes in P.

isfied or not. For *statistical model checking*, this certainty is forfeit. A random set of paths is generated using discrete-event simulation, and these paths are used to make a statement about whether a property is satisfied with a statistically motivated level of confidence. The gain is that some of the problems shared by the numerical algorithms are alleviated. In fact, SMC based on standard (*Monte Carlo*) simulation has none of the three problems stated above. The simulation can be carried out on a higher level than that of the Markov chain, so we do not need to explicitly generate the state space. Furthermore, intertransition times are drawn with knowledge of only the current state so stiffness is also no longer a problem. Finally, Monte Carlo (MC) simulation is trivially extended from Markov chains to Markov reward models or even to non-Markovian transition systems as long as we can draw samples from the transition time distributions (see Chapter 8 of [70]).

However, SMC has well-known challenges of its own.

1. Convergence is slow. To make a simulation result ten times more accurate, the sample size must be increased hundredfold (a quadratic blow-up). This compares very poorly to numerical methods, for which it roughly holds that a fixed number of steps is needed for each digit known with certainty (a logarithmic blow-up). This is particularly a problem for rare events: when the probability of (not) satisfying the relevant property is very small, the accuracy must be high which means that a very large number of runs must be drawn to obtain a reasonable estimate.

2. Choosing the sample size of the simulation experiment is non-trivial. To estimate a small probability a large sample size is needed for the estimate to be accurate. However, since the probability is not known beforehand, it is a priori unclear how large the sample size must be.

3. Dependence on the initial state: while numerical techniques check property satisfaction for the entire state space, statistical techniques require a fixed initial state, which may limit applicability and which means that the verification of nested formulae is no longer possible (although for nested formulae combinations of numerical and statistical techniques may be possible [114]). E.g., to check a property such as $\mathcal{S}_{\bowtie p}(\Phi)$ with Φ non-trivial one needs to know for each state in $\mathcal{X}$ whether this state satisfies Φ, while statistical techniques can only check whether Φ holds for a single state at a time.

4. Termination at the simulation level: termination of the discrete-event simulation is not guaranteed for model checking $\mathcal{P}(\Phi \,\mathsf{U}\, \Phi')$; the simulation termination criterion here is that a state that satisfies $\neg\Phi$ or Φ' is reached, but if the Markov chain is not strongly connected or if it has a state space of infinite size this may happen with probability less than 1.

Throughout this thesis we distinguish between simulation and statistical model checking: by the former we refer to the drawing of individual paths, by the latter we refer to the analysis of a large sample of paths. The four challenges mentioned

above touch one or both of these aspects. For example, Challenge 2 is purely an SMC challenge and Challenge 4 is purely a simulation challenge. However, Challenge 1 touches both; the low probability of observing the relevant behaviour impacts the analysis of the sample but our analysis will focus on changing transition-related probabilities on the simulation level.

In Chapter 2 we discuss the foundations of statistical model checking in more detail and propose new techniques related to Challenge 2. The rest of the chapters focus on Challenge 1 (rare events). The rest of this section is about the common framework underneath these chapters. We will not work towards solutions for Challenges 3 and 4 in this thesis. Particularly, we avoid Challenge 4 by assuming that with probability 1, the simulation terminates in a finite amount of time and we can check whether the property of interest holds on the resulting path within a finite amount of time (see also Assumption 2 of Section 2.1.1).

1.1.5 Model Assumptions

Because statistical model checking depends strongly on an initial state and cannot trivially use nested operators (see Challenge 3 of Section 1.1.4), we can restrict our model setting in a way that allows for easier notation. Let $\vec{x}_0$ be the initial state. From now on, we assume that the model has only two atomic propositions: a and b. The states that satisfy a are called the *tabu* states and the states that satisfy b are called the *goal* states. All states in $\vec{x} \in \mathcal{X}$ are assumed to have $\mathcal{L}(\vec{x}) \in \{\emptyset, \{a\}, \{b\}\}$; if, due to the high-level specification it turns out that a state satisfies both a and b then b gets precedence. In all settings except for the one of Chapter 7, we can assume that all a-states have been merged into a single state x_a and that all b-states have been merged together into a single state x_b.

The path formulae that we are interested in are $\psi \triangleq \neg a \,\mathsf{U}\, b$ and $\psi^{\bar{\tau}} \triangleq \neg a \,\mathsf{U}^{I}\, b$. Many interesting performance measures can be expressed using these probabilities. E.g., the time-bounded *unreliability* can be expressed directly using $\psi^{\bar{\tau}}$, while the Mean Time to Failure (MTTF) and even the steady-state *unavailability* can be rewritten into expressions in which ψ is the only quantity that is difficult to estimate. For $\psi^{\bar{\tau}} = \neg a \,\mathsf{U}^{I}\, b$, we will typically assume that $I = [0, \bar{\tau})$ (i.e., we must reach the b-states before time $\bar{\tau}$), although in Chapter 4 we will also briefly consider the case $I = [\bar{\tau}, \infty)$.

1.2 Monte Carlo Simulation

So far, we have focused on DTMC and CTMC models and on how to specify properties of interest in the languages associated with these models (PCTL and CSL respectively). PCTL and CSL both make assertions about probabilities related to states, and in this section we discuss how to estimate these probabilities using standard (Monte Carlo) simulation. Section 1.2.1 is about how to use sample paths to estimate both path and steady-state probabilities. In Section 1.2.2, we discuss the generation of these sample paths.

1.2.1 Estimating path and steady-state probabilities

We first focus on estimating probabilities of the form $\mathbb{P}(\phi)$, where ϕ is a path formula. The probability that this property is satisfied is given by $\pi_\phi \triangleq \mathbb{P}(\phi) = \mathbb{E}(\mathbf{1}_\phi)$, where $\mathbf{1}_\phi(\omega)$ denotes the indicator function which equals 1 if ω satisfies ϕ and 0 otherwise. For each sample run we can evaluate whether ϕ was satisfied on that run. So, after having sampled a series of runs $\{\omega_1, \ldots, \omega_N\}$ we can estimate π_ϕ using

$$\hat{\pi}_\phi = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_\phi(\omega_i). \tag{1.3}$$

Let $\hat{\sigma}$ be the sample standard deviation of our series of runs, given by

$$\hat{\sigma} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{1}_\phi(\omega_i) - \hat{\pi}_\phi)^2}.$$

The 95%-confidence interval for this estimate is then given by (see [70], §4.5)

$$\left[\; \hat{\pi}_\phi - 1.96 \frac{\hat{\sigma}}{\sqrt{N}} \quad , \quad \hat{\pi}_\phi + 1.96 \frac{\hat{\sigma}}{\sqrt{N}} \; \right]. \tag{1.4}$$

Estimating $v_b \triangleq \lim_{k \to \infty} \mathbb{P}_x(\{\omega : x_k^\omega \vDash b\})$ (the steady-state/limiting probability of being in a b-state) using simulation is a little bit more involved. Since we are interested in the behaviour of the system after a large amount of time in the system has passed, the initialisation of the system is non-trivial. If the system is ergodic (which we typically assume to be the case), we can avoid having to *'warm-up'* the simulation before it reaches approximate equilibrium by applying a renewal argument. In this case, we use the atomic proposition a to define a single state x_a as the *regeneration* state. We then partition the behaviour of the system as time progresses into disjoint *busy cycles*; a busy cycle starts and ends when we enter state x_a. Let Z be the amount of time during which the system is in a b-state during a busy cycle and let D be the duration of a busy cycle. Then $v_b = \mathbb{E}(Z)/\mathbb{E}(D)$. The ratio estimator $\hat{v}_b$ is given by

$$\hat{v}_b = \frac{\hat{z}}{\hat{d}}, \tag{1.5}$$

where $\hat{z}$ and $\hat{d}$ are the Monte Carlo estimates for $\mathbb{E}(Z)$ and $\mathbb{E}(D)$ respectively. This estimator is biased, but strongly consistent (i.e., $\hat{v}_b \to v_b$ as the number of samples goes to infinity; see [70], §9.5.3). We generate different runs for the estimates $\hat{z}$ and $\hat{d}$ to avoid dependence.[8] The 95%-confidence interval (see [70], §9.5.3) is then given by

$$\left[\; \hat{v}_b - 1.96 \frac{\hat{\sigma}_v}{\hat{d}\sqrt{N}} \quad , \quad \hat{v}_b + 1.96 \frac{\hat{\sigma}_v}{\hat{d}\sqrt{N}} \; \right], \tag{1.6}$$

[8] This becomes even more necessary when we use importance sampling because techniques that focus on rare events would lead to a large variance of $\hat{d}$ (more details are given in Section 4.1.3).

where $\hat{\sigma}_v^2 = \hat{\sigma}_z^2 + \hat{\sigma}_d^2 \hat{v}^2$ and $\hat{\sigma}_z^2$ and $\hat{\sigma}_d^2$ are the sample variances of sequences containing the V's and D's respectively.

Although these estimation procedures work in many cases, the downside is that when the probability that we need to estimate is small the number of runs N that we need in (1.3) or (1.5) is enormous. Finding a solution to this problem will be the focus of Section 1.3.

1.2.2 Path Generation

We discuss path generation only at the level of the Petri net.[9] Furthermore, we only discuss the sampling of continuous-time paths, as a discrete-time path is simply a continuous-time path without the transition times.

Let $\mathbf{\Omega}$ be the set of all timed paths as introduced in Section 1.1.2. We generate samples from $\mathbf{\Omega}$ as follows: we start the run at time $\tau_0 = 0$ in state $\vec{x}_0$. We then consecutively determine which transition is fired and how long it takes until this happens. Assume that we have so far observed a path $\boldsymbol{\omega} = (\vec{x}_0^{\boldsymbol{\omega}}, \tau_1^{\boldsymbol{\omega}}, \vec{x}_1^{\boldsymbol{\omega}}, \ldots, \tau_k^{\boldsymbol{\omega}}, \vec{x}_k^{\boldsymbol{\omega}})$. We then determine the next state $X_{k+1}^{\boldsymbol{\omega}}$ by drawing directly from the probability distribution (1.2). The sojourn time in $\vec{x}_k^{\boldsymbol{\omega}}$, given by $\mathcal{T}_{k+1}^{\boldsymbol{\omega}} - \tau_k^{\omega}$, is then determined independently by drawing from the distribution with probability density function

$$f_{\vec{x}_k^{\boldsymbol{\omega}}}(\delta) = \eta(\vec{x}_k^{\boldsymbol{\omega}}) e^{-\eta(\vec{x}_k^{\boldsymbol{\omega}})\delta}, \tag{1.7}$$

using the inversion method [70]. We continue until we can terminate, i.e., satisfy a condition that depends on the property whose validity we seek to evaluate.

We will not further discuss non-Markovian systems in this thesis, but for non-Markovian systems the techniques of Chapter 2 may still be applied. In this setting, one often uses the modelling language of *Generalised Semi-Markov Processes* (GSMPs, see [76] or [42]), a formalism which can be seen as an SPN with non-Markovian probability measures for transitions and sojourn times. The standard simulation procedure is to keep an *event list* $(l_i)_{i=1,\ldots,|T|}$, with $|T|$ the number of transitions in the GSMP. The ith element in the event list corresponds to the next time point at which transition t_i is fired; the next state change is the one corresponding to the firing of the transition t_j with $j = \arg\min_{i=1,\ldots,|T|} \{l_i\}$. One can draw non-Markovian transition times by using techniques such as inversion or accept-reject schemes [70].

1.3 Principles of Importance Sampling

This section will be about our rare event simulation method of choice: Importance Sampling (IS) [50]. If done correctly, IS yields an estimator with lower variance than

[9] If a Petri net level description is not available (as may be the case for the models of Chapter 5), one can view the Markov chain as a Petri net that contains a place for each state, which has transitions for each state transition and which at all times contains only one token, namely in the place corresponding to the current state.

standard MC in situations involving rare events. We first give an intuitive description of IS in Section 1.3.1. We then describe IS formally in Section 1.3.2. In Section 1.3.3 we discuss the variance of IS-estimators and the important properties of Bounded Relative Error (BRE) and Vanishing Relative Error (VRE). In Section 1.3.4 we describe Balanced Failure Biasing (BFB) and forcing, which are commonly used IS-techniques for the kind of reliability models that are typically analysed using model checking. In Section 1.3.5 we discuss the technique of Zero Variance Approximation (ZVA) which we use as the basis for the techniques proposed in the rest of the thesis.

1.3.1 Intuitive Description

Assume that we want to estimate the probability π_ϕ of observing ϕ, and that π_ϕ is very small. Using standard simulation, we randomly draw zeroes and ones such that the fraction of ones is expected to be π_ϕ (see (1.3)). Suppose we now somehow make the probability of drawing a non-zero *twice as large*. Then, if we multiply the value $\mathbf{1}_\phi(\omega_i)$ of the ith run in (1.3) by $\frac{1}{2}$, we obtain an estimator that is unbiased and which has a *lower variance* than the standard estimator. Now suppose we already know π_ϕ and make drawing a non-zero exactly π_ϕ^{-1} times as likely. Hence, we draw a non-zero with probability one and multiply each $\mathbf{1}_\phi(\omega_i)$ by the precise probability that we wish to estimate, resulting in an estimator with *zero variance* (this will be the basis behind the approach of Section 1.3.5).

Hence, we want to apply this principle also in the systems that we consider: to make the drawing of paths that satisfy ϕ more likely. Unfortunately, the systems we study are far too complex to '*just*' multiply the probability of drawing a path that satisfies ϕ by some number and multiply the resulting estimate by a *constant* weighting factor. There are many different paths that satisfy ϕ, and we change the probability of drawing each path by manipulating its individual transition probabilities. This cannot be done naively: if we change a transition probability and let the system remain a Markov chain, then paths that take this transition a different number of times will have different weighting factors. Hence, we need to guarantee that the way we tweak transition probabilities and sojourn time densities does not lead to unexpected results. The basic way to do this in complex stochastic systems will be discussed below.

1.3.2 Basic Setup of Importance Sampling

We first consider the problem of estimating $\mathbb{P}(\psi)$: the probability of hitting a b-state before hitting an a-state. If this probability is low, then there is a strong drift away from b, possibly towards a. The idea is then to increase the probability of taking (firing) transitions that take the system closer to b. This is done using a new probability distribution $\mathbb{Q}$ for the simulation, which we call the *simulation distribution* (this procedure is known as a *change of measure*), such that the transitions toward the b-states are much *more likely* than under the old distribution (1.2). We compensate for

this overestimation by weighting each outcome with the ratio of $\mathbb{P}$ and $\mathbb{Q}$ — like the factor $\frac{1}{2}$ in the example in Section 1.3.1.

Every time a transition is sampled using the new density this weighting factor needs to be considered. The final weighting factor $L_{\mathbb{Q}}$ of a run $\omega = (x_0^\omega, x_1^\omega, \ldots, x_{|\omega|}^\omega)$ is called the *likelihood ratio* and is simply computed as the *product* of the individual ratios in the run, i.e.,

$$L_{\mathbb{Q}}(\omega) = \prod_{i=1}^{|\omega|} \frac{\mathbb{P}(x_{i-1}^\omega \to x_i^\omega)}{\mathbb{Q}(x_{i-1}^\omega \to x_i^\omega)}. \tag{1.8}$$

Our new estimator — replacing (1.3) — then becomes

$$\hat{\pi}_\phi = \frac{1}{N} \sum_{i=1}^{N} L_{\mathbb{Q}}(\omega_i) \cdot \mathbf{1}_\psi(\omega_i). \tag{1.9}$$

It is easy to prove that this estimator is unbiased for any new distribution that assigns positive probability to transitions that have positive probability under the old distribution (by the Radon-Nikodym Theorem, see Chapter 7 of [17]).

We do not need to restrict ourselves to changing the *transition probabilities*. Note that for $\psi^{\bar{\tau}} = \neg a \,\mathsf{U}^{[0,\bar{\tau}]}\, b$, i.e., the probability of visiting a Φ'-state before a $\neg\Phi$-state within $\bar{\tau}$ time units, the system failure probability can also be small because the time interval $[0, \bar{\tau}]$ is too short for a sufficient number of transitions to occur. To remedy this, we can replace the *sojourn time density* f of (1.7) by a new density g with a higher transition rate. If we also account for the ratios f/g in the likelihood ratio $L_{\mathbb{Q}}$ then our estimator remains unbiased and, if done correctly, has an even lower variance. We will use this in Chapters 4 and 5.

1.3.3 Variance of Importance Sampling Estimators

As mentioned in the introduction of this section, we apply IS to obtain an estimator with lower variance than the standard MC-estimator. Since an IS-method is uniquely defined by its associated simulation measure $\mathbb{Q}$, we measure the performance of an IS-method by the variance of $\hat{\pi}_\phi$ under $\mathbb{Q}$, given by

$$\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_\phi) = \mathbb{E}\left(L_{\mathbb{Q}}^2 \cdot \mathbf{1}_\psi\right) - \hat{\pi}_\phi^2.$$

Using $\mathbb{Q} = \mathbb{P}$, we obtain the variance of the MC-estimator: $\hat{\pi}_\phi(1 - \hat{\pi}_\phi)$.

A particularly interesting efficiency metric for an estimator is its *relative error*, given by

$$\frac{\sqrt{\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_\phi)}}{\hat{\pi}_\phi}$$

The relative error of the MC-estimator is given by $\sqrt{(1 - \hat{\pi}_\phi)/\hat{\pi}_\phi}$, which goes to infinity when $\hat{\pi}_\phi$ goes to zero. When the relative error of an estimator does *not* go to infinity when $\hat{\pi}_\phi$ goes to zero, we say that our estimator satisfies the desirable property of *Bounded Relative Error* (BRE, [71]). When it goes to zero, we say that

it satisfies the even more desirable property of *Vanishing Relative Error* (VRE). The BRE-property will be of particular interest in Chapter 5, although it will be mentioned in other parts of the thesis as well.

1.3.4 Failure Biasing and Forcing

One of the classical importance sampling methodologies is *Balanced Failure Biasing* (BFB, [108]); throughout this thesis it will be used as a means of comparison. The main idea underlying BFB is that all transitions in the SPN can be divided into two categories: transitions that bring the system *closer* to the goal state(s) and transitions that take the system *away* from the goal state(s). This is a natural assumption in models of highly reliable multicomponent systems, where the transitions are either component failures or repairs and the goal states are states in which certain (configurations of) components are broken. The probability of interest in this setting is small because failure transitions are much less likely than repair transitions. The idea is then to choose a number $c \in (0,1)$ such that the total probability of a failure transition is always c. Specifically, if in a state $\vec{x} \in \mathcal{X}$ it holds that $m > 0$ failure transitions and $n > 0$ repair transitions are enabled, then BFB uses the new measure $\mathbb{Q}$ given by

$$q_{\vec{x}}(j) = \begin{cases} \dfrac{c}{m} & \text{if } t_j \text{ is a failure transition,} \\ \dfrac{1-c}{n} & \text{if } t_j \text{ is a repair transition.} \end{cases} \tag{1.10}$$

A typical choice for c is $\frac{1}{2}$.

In a highly reliable system, it might also be that $\mathbb{P}(\psi^{\bar{\tau}}) = \neg a \,\mathsf{U}^{[0,\bar{\tau}]}\, b$ is small because failures occur slowly, i.e., by the time the system has failed the time bound $\bar{\tau}$ has long been passed. In highly reliable systems, this is usually because it takes a long time until the next failure occurs when all components all working. The idea is then to apply *forcing* (see [80] or [81]): we draw transition times *conditional* on their occurrence before the time bound $\bar{\tau}$.[10] In particular, if we are in state $\vec{x}$ at time τ we draw from the alternative density

$$g_{\vec{x},\bar{\tau}-\tau}(\delta) = \frac{\eta(\vec{x})e^{-\eta(\vec{x})\delta}}{1-e^{-\eta(\vec{x})(\bar{\tau}-\tau)}}, \quad \delta \in [0, \bar{\tau}-\tau]. \tag{1.11}$$

1.3.5 Zero Variance

Consider the following ideal situation: for every state $\vec{x}$ and for all time points $\tau \in [0,\bar{\tau}]$ we already *know* the probability $\pi_\phi(\vec{x},\tau)$ of seeing $\psi^{\bar{\tau}}$, i.e., seeing a b-state before a a-state within $\bar{\tau}-\tau$ time units. Let $\vec{x}+u_j$ be the new state that we obtain

[10] This is not to be confused with *conditioning* or *discrete time conversion* [41, 45] in which we numerically compute the conditional probability $\mathbb{P}(\psi^{\bar{\tau}}|\boldsymbol{\omega})$, where $\boldsymbol{\omega}$ is a timed path in which transition times have *not* been drawn for some or all of the states.

if transition j is chosen when we are in state $\vec{x}$. Then we can introduce a new simultaneous density of the transition $t_j \in T$ and sojourn time $\delta \in [0, t-\bar{\tau}]$, namely

$$q_{\vec{x}}(j,\delta) = \frac{p_{\vec{x}}(j) \cdot f_{\vec{x}}(\delta) \cdot \pi_\phi(\vec{x}+u_j, \tau+\delta)}{\int_0^{\bar{\tau}-\tau} \sum_{j'=1}^{|T|} p_{\vec{x}}(j) \cdot f_{\vec{x}}(\delta) \cdot \pi_\phi(\vec{x}+u_{j'}, \tau+\delta') d\delta'}. \tag{1.12}$$

In the timeless setting, (1.12) simplifies to

$$q_{\vec{x}}(j) = \frac{p_{\vec{x}}(j) \cdot \pi_\phi(\vec{x}+u_j)}{\sum_{j'=1}^{|T|} p_{\vec{x}}(j') \cdot \pi_\phi(\vec{x}+u_{j'})}, \tag{1.13}$$

This new simulation density is proven to yield an estimator with zero-variance in [31]. Of course, we do not explicitly know π_ϕ; if we would know, we would not need to simulate at all. However, we might be able to come up with an approximation for π_ϕ. We denote this approximation by w. Then, we replace the function π_ϕ in (1.12) by w. This approach is called *Zero Variance Approximation* (ZVA). If the simulation distribution associated with the approximation w is good enough then we have succeeded in overcoming the main problem facing standard Monte Carlo simulation of rare events. Chapters 4, 5 and 7 will all be about finding approximations that lead to well-performing simulation measures.

1.4 Contributions of this Thesis

The contributions of this thesis are as follows.

In **Chapter 2**, we present an overview of the known hypothesis tests from the statistical model checking literature and cast them into a single framework. We demonstrate that the performance of these known methods is very sensitive to parameters that must be set *a priori*. We introduce two new methods for which the probability of drawing the right conclusion can be bounded from below, arbitrarily close to 1, regardless of how little is known about the system beforehand. We conduct a case study in which we compare the correctness and the efficiency of the two new methods and the earlier methods; the results confirm the robustness of the new methods with respect to their parameters.

The contents of Chapter 2 are published in the following paper, of which a journal version is pending.

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "On Hypothesis Testing for Statistical Model Checking," SMC Workshop 2013.

In **Chapters 3 and 4**, we consider (parallel networks of) birth-death processes, and discern several asymptotic regimes that all lead to different typical behaviour to reach the rare event. We zoom in on the regime where the probability of observing a large population is small due to the high speed of the deaths, and present a method based on importance sampling to speed up the simulation process in this case. We

conduct a case study based on a well-known literature benchmark involving system failure in a multicomponent system (in this case, births corresponds to component failures and deaths to repairs).

The contents of Chapters 3 and 4 have been published in the following papers:

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," in *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, 2010, pp. 251–260.

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," *Performance Evaluation*, vol. 69, no. 7, pp. 336–355, 2012.

- D. Reijsbergen, P.T. de Boer, and W. Scheinhardt, "Transient behaviour in highly dependable Markovian systems: New regimes, multiple paths," RESIM 2010.

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Fast simulation for slow paths in Markov models," RESIM 2012.

In **Chapter 5**, we present an algorithm for importance sampling in general Markov chains that is guaranteed to produce an estimator that meets the conditions presented in [73] [72] for vanishing relative error. Although our method works on the level of the state space, it does not suffer from high-probability cycles in the model and it can handle infinite state spaces under certain conditions.

In **Chapter 6** we demonstrate how the procedure that is used to obtain the change of measure in Chapter 5 can be executed a second time to achieve even further variance reduction, using ideas from [59], and also apply this technique to the method of failure biasing, with which we compare our results.

The contents of Chapters 5 and 6 have been published in the following paper (a journal version is pending):

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and S. Juneja, "Some advances in importance sampling of reliability models based on zero variance approximation," RESIM 2012.

In **Chapter 7** we present a formal algorithm that obtains the information required to construct a good change of measure from a high-level SPN-description, without generating the full state space. Essentially, the algorithm of this chapter reduces the state space of the model into a (much smaller) graph in which each node represents a set of states for which the most likely path to failure has the same form. We empirically demonstrate the efficiency of the method with two case studies.

The contents of Chapter 7 have been published in the following paper (a journal version is pending):

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Automated rare event simulation for stochastic petri nets," in *Proceedings of the 10th International Conference on the Quantitative Evaluation of Systems (QEST)*, 2013.

Chapter 8 concludes the thesis.

CHAPTER 2

Sequential Hypothesis Testing

As mentioned in the introduction, the main theme of this thesis is to validate a formal property in a probabilistic system using a sample of simulation runs as efficiently as possible. Typically, these formal properties are (state) properties which are specified in terms of a bound on some probability of interest (see Section 1.1.2); we denote this probability of interest by π. The exact way in which the simulation runs are then interpreted depends on the interests of the investigator. First of all, she could be interested in a *quantitative* statement, consisting of an estimate of the performance measure with a corresponding *confidence interval* (e.g., with 95% confidence, the probability of deadlock before termination is 10% with a 2% margin of error). Secondly, she could be interested in a *qualitative* statement about a performance property, specified as a *hypothesis* that asserts that the true probability π is larger (or smaller) than some boundary value π_0 (e.g., with 95% confidence, the probability of deadlock before termination is greater than 5%). While Chapters 4 to 7 are primarily concerned with quantitative statements in the context of rare events, the focus of this chapter will be on qualitative statements in general.

The question of how to perform a qualitative test for statistical model checking in a methodologically sound way has been the subject of debate for the past few years. Seminal in the field of establishing a formal framework was the work of Younes [113, 115, 116], who used a sequential sampling scheme introduced by Wald [111]. Here '*sequential*' means that after generating each sample it is checked whether we can conclude the test. While Wald's test was originally not suited for probabilistic model checking problems, Younes showed that if the true probability π was assumed not to lie inside a (small) *indifference region* $[\pi_0 - \delta, \pi_0 + \delta]$, Wald's test can be recast into the statistical model checking framework.

The most common alternatives to Wald's test are those based on a confidence interval (CI) constructed using a *fixed number of samples*. In order to determine the fixed sample size, the investigator can use a *guess for* π, on which she bases (through approximate normality based on the Central Limit Theorem (CLT) or a bound such as the Chernoff-Hoeffding bound) a *guess for the number of samples needed*.

Given this selection of tests, it is then up to the investigator to decide which test she finds the most appealing. There are three main criteria by which to judge the appeal of these tests.

1. *Correctness*, which asserts that the probability of rejecting a valid hypothesis

is guaranteed to be smaller than 1 minus the given confidence level,

2. the *power*, which is the probability of rejecting an invalid hypothesis,[11] and

3. the *efficiency*, the average number of samples needed before a conclusion can be drawn.

The guess underlying the aforementioned fixed sample size test determines the power of the test but not its correctness. Meanwhile, the correctness of Wald's sequential test depends on the indifference level δ while its power does not. In some settings, the investigator may indicate beforehand that she no longer cares about the validity of the test when the true parameter π is close enough to the bound π_0. In such a situation, the choice of indifference level δ emerges naturally. If this is *not* the case, the investigator will generally not have a clear idea about its value, which may result in a choice of indifference level δ that is far too small. This can be *detrimental* to the efficiency of the test. For Wald's sequential test, it holds that if an indifference level is made ten times smaller, it will on average take about ten times longer to draw a conclusion. On the other hand, if the indifference level is chosen too large, the outcome of the test may get arbitrarily close to flipping a single coin. The problem here is that when a conclusion is drawn, the investigator *cannot* decide, based on the outcome of the test, whether it was carried out properly or whether a slightly biased coin was flipped. The investigator can only accept the outcome, having good faith that her assumptions were satisfied.

Of course, all tests mentioned so far have proven their merit in recent years. However, we argue in this chapter that when the investigator wants strong guarantees in terms of correctness and power that do not depend on parameters, her demands are not met by the tests currently available in the statistical model checking literature. As such, we introduce two sequential tests for statistical model checking, one based on our own analysis and one based on [28]. Only the efficiency (and not the correctness or power) of these two tests depends on a guess. The second test is very insensitive to the guess, and is very well suited for situations in which the investigator really has no idea about the true probability π, at least not up to two orders of magnitude. The first test is superior when the guess is actually close to the true probability π, but the investigator does not want to commit herself to an indifference level — if the guess is far off, then this test will simply require an enormous amount of samples, whereas Wald's test will in this case accept the wrong hypothesis with high probability without giving any indication that the test went wrong.

The contributions of this chapter are the following. We present a single framework that allows the tests discussed in this thesis to be compared in a clear manner in Section 2.1. In Section 2.2, we present an overview of the main tests proposed so far for statistical model checking, using the framework of Section 2.1. We construct a new test for statistical model checking based on original analysis, inspired

[11] This will be the definition for the power that we use in this thesis. In [28], it is defined as the probability of rejecting the *null hypothesis* (see Section 2.1) when it is invalid.

by Ross' Generalized Azuma Inequality, and introduce one of the tests of [28] to statistical model checking in Section 2.3. We compare the performance of all these tests empirically in Section 2.4. We discuss the possibility of a sequential test for model checking steady-state properties based on the analysis of the two new tests in Section 2.5. Section 2.6 concludes the chapter.

2.1 General Framework

In this section we will discuss the framework that we will use to compare the tests described in Sections 2.2 and 2.3. We begin in Section 2.1.1 with some notes on the generality of the framework and its description. In Section 2.1.2 we discuss elementary statistical methodology in order to fix terminology and notation, before we move on the description of the three hypotheses and the corresponding testing procedure used in this chapter.

2.1.1 Assumptions

In this section we focus on verifying the property $\mathcal{P}_{>\pi_0}(\phi)$, which asserts that the probability of observing a path satisfying ϕ from the initial state is greater than π_0.[12] Consequently, the simulation procedure under consideration consists of drawing sample paths and verifying whether ϕ holds on each of these path. Steady-state properties will be discussed in Section 2.5.

While we focused on Markovian systems and the associated logics PCTL and CSL in Chapter 1, the statistical approach discussed in this chapter is more general — the only requirement on the system model is that we can randomly generate execution paths for which we can check whether some path property ϕ is satisfied. This can be rewritten into the following requirements:

1. we can generate execution paths from the model, according to a well-defined probability measure on the execution paths;

2. with probability 1, these paths are generated in a finite amount of time and we can test, also in a finite amount of time, whether the property ϕ holds on a path;

3. we do not encounter nondeterminism [10], or we at least have a well-defined policy or scheduler to resolve it.

In principle, we do not need *any* additional information about the system model as long as we can obtain execution paths that satisfy these three requirements. We commonly call a system about which additional information is indeed not available

[12] By treating $\mathcal{P}_{>p_0}(\phi)$, we treat, without loss of generality, all possible variations of the probabilistic path operator $\mathcal{P}$, because, using statistical model checking, we cannot differentiate between $\mathcal{P}_{>p_0}(\phi)$ and $\mathcal{P}_{\geq p_0}(\phi)$ (more on that later in Section 2.1), and because $\mathcal{P}_{<p_0}(\phi)$ and $\mathcal{P}_{\geq 1-p_0}(\neg\phi)$ are equivalent.

a *black-box* system [106], and the methodology described in Section 2.1.2 already works for these systems.

In practice, a system model is often available that allows us to write a computer program that can generate execution paths, so that the system is not completely black-box. Popular modelling formalisms include Generalized Semi-Markov Processes (GSPMs, [42, 76]) and stochastic (possibly *non-Markovian*) Petri nets [47]. Requirements 2) and 3) will not be satisfied in all GSMPs or stochastic Petri nets, and even if they are not, it might still be possible to apply a refined form of statistical model checking (see, for example, [37], [107] or [112], in which 2) is not satisfied). Judging whether requirements 2) and 3) are satisfied given a system model and performance property is a field of study in itself. As this chapter is not about generating sample paths but about the interpretation of the results, we refer the interested reader to the vast literature on stochastic simulation [40, 97], and from now on assume that we draw samples from a black-box system in order to say something about $\mathcal{P}_{>\pi_0}(\phi)$.

In this chapter, we will not consider nested probabilistic operators. To read about how nested operators are treated in other settings, see, e.g., Section 3.2 of [107] or [114], in which a combined numerical/statistical procedure is proposed.

2.1.2 Statistical Framework

Let ω_i be the execution path in the i-th sample, $i = 1, \ldots, N$, and define

$$X_i \triangleq \mathbf{1}_\phi(\omega_i) = \begin{cases} 1 & \text{if } \phi \text{ holds on } \omega_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Then X_i has a Bernoulli distribution with parameter π_ϕ, where π_ϕ denotes the true probability that ϕ is satisfied. Writing π for π_ϕ, this means that

$$\mathbb{P}(X_i = x) = \begin{cases} \pi & \text{if } x = 1, \\ 1 - \pi & \text{if } x = 0. \end{cases}$$

The total sample $\vec{X} \triangleq (X_i)_{i=1,\ldots,N}$ will be used to perform a statistical *test*. To do this, we combine all relevant information from the individual sample paths into a function that maps $\{0,1\}^N$ onto $\mathbb{R}$, called the *test statistic*. We use the test statistic to falsify claims about π, called *hypotheses*. If we can show that, under the condition that some hypothesis H is true, the probability that the observed outcome of the test statistic occurs is smaller than some given $\alpha \in (0, \frac{1}{2})$, then we *reject* H. The parameter α is called the *significance*, and $1 - \alpha$ is called the *confidence* of the test. A hypothesis that can be rejected this way is called a *null hypothesis*, while a hypothesis that can be accepted through the rejection of a null hypothesis is called an *alternative hypothesis*. Rejecting a valid null hypothesis is called an *error of the first type* (or a *false positive*). Not accepting a valid alternative hypothesis is called an *error of the second type* (or a *false negative*).

Since we are interested in checking whether $\mathcal{P}_{>\pi_0}(\phi)$ holds, there are two relevant claims: $\pi > \pi_0$ and $\pi \leq \pi_0$. There is no clear distinction between a null and

alternative hypothesis, as there is no asymmetry in our desire to reject any one of the two claims. Accordingly, we specify *two* alternative hypotheses, each of which we would like to accept if it were true:

$$\begin{aligned} H_{+1} &: \pi > \pi_0, \\ H_{-1} &: \pi < \pi_0. \end{aligned} \tag{2.2}$$

Additionally, we have the null hypothesis

$$H_0 : \pi = \pi_0.$$

Note the null hypothesis *cannot* be shown to be correct, as its negation $\pi \neq \pi_0$ cannot be disproved statistically. The reason is that no matter how many samples we draw and no matter how much evidence we see for $\pi = \pi_0$, there will always be some small ϵ such that we cannot reject the claim that $\pi = \pi_0 + \epsilon$.

The procedure to test which of the alternative hypotheses is true is as follows: after having drawn N samples, we let $S_N(\vec{X})$ be the test statistic given by the sum of X_1 up to X_N, and omit the argument $\vec{X}$ for brevity. We can then view the evolution of S_N as the evolution of a discrete-time Markov chain on state space $\mathbb{N}$, with the number of drawn samples on the x-axis and the value of the test statistic on the y-axis, where in each step we take a jump to the right or top-right (as can be seen in Figure 2.1).

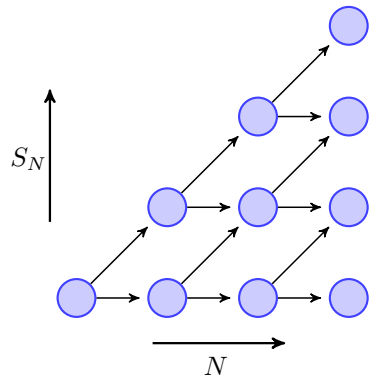


Figure 2.1: Markov chain representation of a hypothesis test.

While we are drawing samples, the expected behaviour of the process S_N is that it drifts away from the x-axis. The true parameter π determines the speed of this drift. Remember that our main interest is to test whether $\pi - \pi_0$ is positive or negative. Hence, we focus on the *shifted* test statistic

$$Z_N \triangleq S_N - N\pi_0.$$

The speed at which Z_N drifts away from the x-axis is completely determined by $\pi - \pi_0$. If $Z_N \gg 0$ then this is strong evidence for H_{+1}, while if $Z_N \ll 0$ then this is strong evidence for H_{-1}.

We then specify *four* test decision areas which are subsets of $\mathbb{R}^2$. Three of them are *critical*, by which we mean that we draw a conclusion as soon as they are entered by Z_N. The first critical area $\mathcal{U}$ is the area such that as soon as Z_N enters $\mathcal{U}$, we accept H_{+1}. The second critical area $\mathcal{L}$ does the same for H_{-1}. As soon as Z_N enters the critical area $\mathcal{I}$, we stop the test without accepting any hypothesis. We accordingly say that the test was *inconclusive*. All that is outside these three areas makes up the *non-critical* area $\mathcal{NC}$.

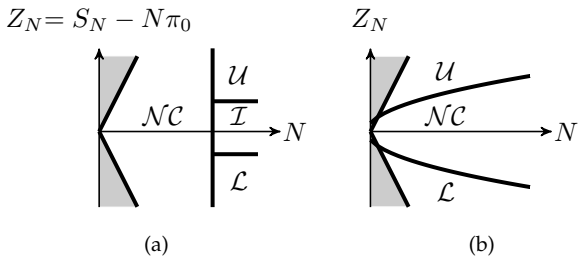


Figure 2.2: Graphical representation of the test decision areas $\mathcal{L}$, $\mathcal{U}$, $\mathcal{I}$ and $\mathcal{NC}$. Left: fixed sample size test. Right: sequential test. Grey areas represent areas in which Z_N cannot go.

The tests that we consider in this chapter are completely determined by the shape of these areas. In Figure 2.2, we display examples of these regions for both a typical fixed sample size and a typical sequential test. Note that sequential tests in principle do not have an area $\mathcal{I}$, since we keep sampling until we draw a conclusion. In practice, we might set a time-out parameter τ, letting all states that are to the right of τ be part of $\mathcal{I}$.

Given $\mathcal{L}, \mathcal{U}$ and $\mathcal{I}$, we want to bound the probability of entering them given that a hypothesis is valid. Let A_i be the event that we accept H_i. Then we impose the following two conditions on the two errors of the first type:

$$\mathbb{P}(A_{+1} \mid \neg H_{+1}) \leq \alpha_1, \tag{2.3}$$
$$\mathbb{P}(A_{-1} \mid \neg H_{-1}) \leq \alpha_2, \tag{2.4}$$

and we impose the following conditions on the two errors of the second type:

$$\mathbb{P}(\neg A_{+1} \mid H_{+1}) \leq \beta_1, \tag{2.5}$$
$$\mathbb{P}(\neg A_{-1} \mid H_{-1}) \leq \beta_2. \tag{2.6}$$

Throughout this chapter, we will choose $\alpha_1 = \alpha_2$ and $\beta_1 = \beta_2$. In principle, the total probability of error of the first type should always be set to $\alpha \triangleq \alpha_1 + \alpha_2$, since under H_0 both errors may occur. However, we assume that H_0 does not hold, as we argued earlier in this section. So in this case, we can set $\alpha = \alpha_1 = \alpha_2$ and achieve the same level of confidence with a smaller sample size. Also when $H_0 : \pi = \pi_0$ *is* possible (as, e.g., in the case of a 'simple' underlying model and performance

property), the investigator can explicitly decide that she does not *care* about the validity of the results when π equals or is very close to π_0, so that it is justifiable for her to say that the total probability of error is $\alpha_1 = \alpha_2$.

A test is called *optimal* if there does not exist a test with the same (or better) error bounds that on average needs fewer samples to come to a conclusion. However, the tests that we discuss in this chapter satisfy bounds on their error probabilities that are fundamentally different from each other, so we do not further speak of optimality. Discussing known tests will be the focus of Section 2.2, while discussing the new tests will be the topic of Section 2.3.

2.2 Known Statistical Hypothesis Tests

In this section, we discuss the four tests most commonly found in model checking tools and the literature. Each of them can be seen as a special case of the general framework presented in Section 2.1. In Section 2.2.1 we discuss a test based on the the (standard) binomial test for which we draw an a priori fixed number of samples which we from now on will call the '*Gauss*' test. In Section 2.2.2 we discuss a *sequential* test, i.e., a test in which we check after each individual sample whether a conclusion can be drawn, called the '*Sequential Probability Ratio Test (SPRT)*' — this is the test of Younes and Wald as we discussed in the introduction to this chapter. In Section 2.2.3, we discuss a fixed sample size test based on probabilistic principles different from those of the Gauss test, that we call the '*Chernoff*' test. In Section 2.2.4, we will discuss the Bayesian version of the SPRT, which we call the '*Bayes*' test. Later, we will empirically compare the performance of these tests to the two tests of Section 2.3.

2.2.1 Gauss Test

First, we assume that N, the number of samples to be drawn, is fixed beforehand. The sets $\mathcal{U}, \mathcal{L}$ and $\mathcal{I}$ then only contain points (x, n) for which $n \geq N$, as in Figure 2.2a. We accept $H_{+1} : \pi > \pi_0$ when Z_N is large and positive, we accept $H_{-1} : \pi < \pi_0$ when Z_N is large and negative and dismiss the test as inconclusive when Z_N is close to 0. In fact, the whole test can be completely characterised by a function $u(N)$ which defines the boundary between $\mathcal{U}$ and $\mathcal{I}$ and a function $l(N)$ which does the same for $\mathcal{L}$ and $\mathcal{I}$.

So, $u(N)$ must now be chosen such that when H_0 or H_{-1} is true, the probability that $Z_N > u(N)$ is smaller than α. A fundamental insight is that it is sufficient to check whether the probability that $Z_N > u(N)$ under just H_0 is less than α. The reason is that under H_{-1}, high values of Z_N are even less likely than under H_0, and if the probability of $Z_N > u(N)$ under H_0 is already smaller than α, then this certainly also holds under H_{-1}, i.e.,

$$\mathbb{P}(Z_N > u(N)|H_{-1}) \;<\; \mathbb{P}(Z_N > u(N)|H_0) \;=\; \mathbb{P}(Z_N > u(N)|\pi = \pi_0)$$

Analogously, we base $l(N)$ only on H_0 and not on H_{+1}. Assuming H_0 is true, the individual samples X_i, $1 \leq i \leq N$, have a Bernoulli distribution with parameter π_0, which means that the test statistic S_N has a binomial distribution with parameter N and π_0. This means that we accept H_{+1} if

$$\sum_{k=0}^{S_N} \binom{N}{k} \pi_0^k (1-\pi_0)^{N-k} \geq 1 - \frac{\alpha}{2}. \tag{2.7}$$

For small N the exact binomial distribution is still computationally easy. For large values of N this is less so, but the Central Limit Theorem (CLT) tells us that the binomial distribution found on the left side of (2.7) can be well approximated by a normal distribution: let[13]

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\boldsymbol{\pi}}} e^{-z^2/2} dz, \tag{2.8}$$

then we accept H_{+1} if

$$\Phi\left(\frac{S_N - N\pi_0}{\sqrt{N\pi_0(1-\pi_0)}}\right) = \Phi\left(\frac{Z_N}{\sqrt{N\pi_0(1-\pi_0)}}\right) \leq 1 - \frac{\alpha}{2}.$$

As a result, we find (after making a similar argument for l) that

$$l(N) = \Phi^{-1}\left(\frac{\alpha}{2}\right)\sqrt{N\pi_0(1-\pi_0)}, \tag{2.9}$$

$$u(N) = \Phi^{-1}\left(1-\frac{\alpha}{2}\right)\sqrt{N\pi_0(1-\pi_0)} = -l(N). \tag{2.10}$$

Throughout the rest of this thesis we call the test of this section the '*Gauss*' test because of the approximation based on the CLT. In the case study of Section 2.4, we always use this approximation, also for 'small' values of N. In [37], tests of this form (the authors of this paper use the exact binomial distribution) are called Single Sampling Plan (SSP) tests.

The astute reader will notice the relation between the values of l and u on one hand and the borders of the confidence interval of level $1-\alpha$ (the $(1-\alpha)$-CI) for $(\pi - \pi_0)N$ on the other hand — they are equal if π_0 in (2.9) and (2.10) is replaced by Z_N/N. Confidence intervals could be used for the following procedure: draw N samples, construct the $(1-\alpha)$-CI around Z_N and reject the null hypothesis if 0 is not inside it. If so, accept H_{+1} if $Z_N > 0$ and H_{-1} otherwise. Using the preceding arguments one can argue that this procedure is similar to the one described above, and if N is large (which we assume in order to use the CLT) the difference is negligible. As a side remark, a confidence interval constructed using the exact binomial distribution is often called a '*Clopper-Pearson interval*' in the scientific literature.

The construction of a CI for π is implemented in several model checkers, e.g., in PRISM and UPPAAL. By the same principles, a hypothesis test can be carried out by hand by checking whether π_0 is contained in this CI or not.

[13] Note that the $\boldsymbol{\pi}$ used in (2.8) refers to the mathematical constant and not to the probability of interest.

The main argument against the use of this method is its dependence on the choice of N. If π is very different from π_0, a small value for N suffices — choosing N too large then leads to extra inefficiency. Alternatively, if π is close to π_0 a large value for N is needed — choosing N too small then leads to a decrease in power. In [64], it was proposed to keep sampling until the CI had reached a prespecified width (this is implemented in the tool MRMC [63]). However, this method does not yield clear bounds on the error probabilities of the second type. Instead, we focus on choosing N such that (2.5) and (2.6) are satisfied, as we will describe in the following.

If $\pi - \pi_0$ were known to be equal to some given value ϵ, then the minimal choice of N that still satisfies (2.5) and (2.6) can be calculated. If N is large enough, then $\hat{\pi}_N \triangleq S_N/N$ approximately has a normal distribution with mean $\pi_0+\epsilon$ and variance $\sigma^2 \triangleq (\pi_0 + \epsilon)\,(1 - \pi_0 - \epsilon)/N$. Writing $\xi = \Phi^{-1}(1 - \frac{\alpha}{2})$ and $\sigma_{H_0}^2 \triangleq (\pi_0)\,(1 - \pi_0)/N$, the probability of *not* being able to reject H_0 after drawing N samples is given by

$$\begin{aligned}
&\mathbb{P}\left(\pi_0 \in [\hat{\pi}_N - \xi\sigma_{H_0}, \hat{\pi}_N + \xi\sigma_{H_0}]\right) \\
&= \mathbb{P}\left(-\xi\sigma_{H_0} \leq \hat{\pi}_N - \pi_0 \leq \xi\sigma_{H_0}\right) \\
&= \mathbb{P}\left(\frac{-\xi\sigma_{H_0} - \epsilon}{\sigma} \leq \frac{\hat{\pi}_N - \pi_0 - \epsilon}{\sigma} \leq \frac{\xi\sigma_{H_0} - \epsilon}{\sigma}\right) \\
&= \Phi\left(\frac{\xi\sigma_{H_0} - \epsilon}{\sigma}\right) - \Phi\left(\frac{-\xi\sigma_{H_0} - \epsilon}{\sigma}\right).
\end{aligned} \tag{2.11}$$

Setting this equation equal to some value β and solving numerically for N guarantees that if the true value π is indeed equal to $\pi_0 + \epsilon$, then the probability of an error of the second type is at most β. This gives a good indication about how many samples are needed, as we will show in Section 2.4. In reality we do not know ϵ, but we may be able to come up with a *guess* which we can substitute for ϵ into (2.11).

If the test terminates as inconclusive after having drawn N samples, a common pitfall is to draw a second batch of N samples and check if a hypothesis can be rejected based on the total of $2N$ samples. However, if there was a probability α of an error of the first type during *both* subexperiments, the probability of error during *any one* of the two subexperiments is at worst equal to $1 - (1 - \alpha)^2 \approx 2\alpha$, so that (2.3) and (2.4) are not satisfied. Of course, one can choose the new confidence level for the individual tests α' such that α equals $1 - (1 - \alpha')^2$ but then one could wonder why one would run only two tests instead of any other number of tests.

In the most extreme application of this idea one would check *after each* sample n whether a conclusion can be drawn — we call such a test a *sequential* test. Clearly, the requirements (2.3) and (2.4) on the errors of the first type are again not satisfied. The subject of the following section will be a sequential test that does satisfy (2.3) and (2.4), namely the SPRT.

2.2.2 Sequential Probability Ratio Test (SPRT)

The SPRT for statistical model checking was introduced by Younes[14] in [115], based on ideas that go back to [111]. In [111], Wald tries to sequentially test which of the following two hypotheses is true,

$$\begin{array}{lcl} H_{+1} & : & \pi \geq \pi_{+1}, \\ H_{-1} & : & \pi \leq \pi_{-1} \end{array} \tag{2.12}$$

for values $\pi_{-1} < \pi_{+1}$. He argues that a suitable test statistic is the so-called hypotheses' likelihood ratio:

$$T_N \triangleq \frac{\pi_{+1}^{S_N}(1-\pi_{+1})^{N-S_N}}{\pi_{-1}^{S_N}(1-\pi_{-1})^{N-S_N}}.$$

Clearly, small values of T_N speak in favour of H_{-1} while large values speak for H_{+1}. The idea is then to construct boundaries l' and u' such that if T_N becomes larger than u' we accept H_{+1} and if T_N gets below l' we accept H_{-1}. We then have to bound, for given boundaries $l' < u'$, the probability of crossing l' given H_{+1} and the probability of crossing u' given H_{-1}. Wald showed that this is indeed possible. In particular, for $l' = \beta/(1-\alpha)$ and $u' = (1-\beta)/\alpha$ one knows that the probability of accepting H_{-1} while H_{+1} is true is smaller than α while the probability of accepting H_{+1} while H_{-1} is true is smaller than β.

To evaluate the validity of $\mathcal{P}_{>\pi_0}(\phi)$, we have the hypotheses of (2.2), which are similar to those of (2.12) with $\pi_{+1} = \pi_{-1} = \pi_0$. Unfortunately, in this case the value T_N is always 1. The idea proposed in [115] is to choose an *indifference* level δ such that we can safely assume that the true value for π is not inside the interval $[\pi_0 - \delta, \pi_0 + \delta]$. Then we can set $\pi_{-1} = \pi_0 - \delta$ and $\pi_{+1} = \pi_0 + \delta$ and carry out the above procedure.

This approach is closely related to the fixed sample size procedure of Section 2.2.1. To better understand the relationship, first note that instead of the test statistic T_N we could also use

$$\log T_N \triangleq q_1 S_N + q_2 N,$$

where

$$q_1 = \log\left(\frac{p_{+1}\cdot(1-\pi_{-1})}{(1-\pi_{+1})\cdot p_{-1}}\right),\ q_2 = \log\left(\frac{1-\pi_{+1}}{1-\pi_{-1}}\right).$$

Hence, an equivalent formulation is to use the process $Z_N = S_N - N\pi_0$ of Figure 2.2 as a test statistic, with boundaries

$$l(N) = \frac{1}{q_1}(\log l' - q_2 N) - N\pi_0, \tag{2.13}$$

$$u(N) = \frac{1}{q_1}(\log u' - q_2 N) - N\pi_0. \tag{2.14}$$

[14] We use slightly different terminology than the authors of [115]. They use H_0 for H_{+1} of (2.12), they use H_1 for the H_{-1} of (2.12), and they use H_2 to denote $\pi \in [\pi_{-1}, \pi_{+1}]$. Furthermore, where they speak of a type 1 error and a type 2 error in the case of the SPRT, we speak of two errors of the first type.

These are linear functions in N. So whereas the boundaries of (2.9) and (2.10) are proportional to $\sqrt{N}$, the boundaries of (2.13) and (2.14) increase linearly. One can verify that when $\pi_0 = \frac{1}{2}$ or in the limit $\delta \downarrow 0$ the boundaries are constants.

In Figure 2.3, we display a possible run of Z_N for $\pi_0 = \frac{1}{3}$. The two solid lines represent $l(N)$ and $u(N)$. They tilt slightly downward, although this is not easily visible at this scale. The dotted lines represent the expected evolution of Z_N if π is exactly equal to $\pi_0 - \delta$ or $\pi_0 + \delta$. Typically, if π is truly outside the indifference region, the up- or downward tilt of l and u is barely noticeable.

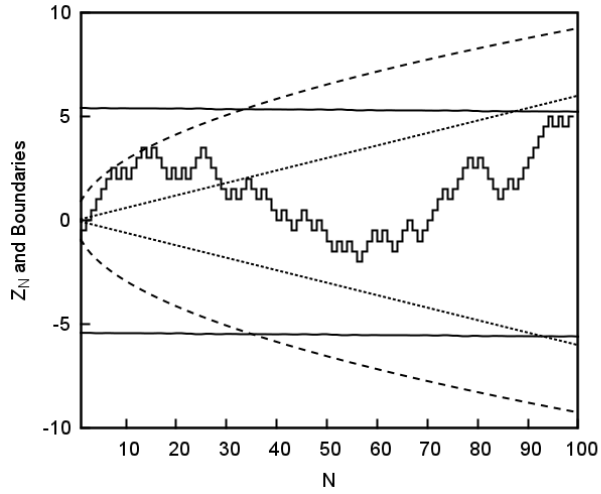


Figure 2.3: Run and test decision area boundaries when $\pi_0 = \frac{1}{3}$, $\alpha = \frac{1}{20}$, $\delta = \frac{3}{50}$. Step function: realisation of Z_N with $\pi = \pi_0$. Solid lines: area $\mathcal{NC}$ boundaries for the SPRT. Dashed lines: area $\mathcal{NC}$ boundaries for the Gauss test seen as a function of the fixed sample size N. Dotted lines: expected trajectories of Z_N if $\pi = \pi_0 - \delta$ and $\pi = \pi_0 + \delta$ respectively.

The remaining question is then how to choose δ. In some settings, the parameter δ arises naturally. For example, if the investigator explicitly states that she does not care if the true probability π is 5.01% instead of 5%, then $\delta = 0.0001$ would be a natural parameter choice. If the choice of δ does *not* arise naturally, then the dependence on δ is an argument against the use of the SPRT. The probabilities of error of the first type are completely determined by δ, being equal to the bounds when $\delta = |\pi - \pi_0|$. Since $|\pi - \pi_0|$ is unknown, picking δ is as hard as solving the model checking problem itself, and picking δ too conservatively will cause a large increase in the expected amount of time before the test can be concluded.

2.2.3 Chernoff Test

In [53], Hoeffding uses an application of earlier results by Chernoff to establish a bound on the probability that a sum of independent random variables exceeds some threshold value. Known as the Chernoff-Hoeffding bound or simply as Hoeffding's inequality, it states that for any sequence $X_1, X_2, \ldots, X_N$ of independent random variables with $\mathbb{P}(0 \leq X_i \leq 1) = 1$, it holds for all $t > 0$ that

$$\mathbb{P}(\bar{X} - \mathbb{E}(\bar{X}) > t) \leq e^{-2Nt^2}, \tag{2.15}$$

where $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$, and as a corollary it follows that

$$\mathbb{P}(|\bar{X} - \mathbb{E}(\bar{X})| > t) \leq 2e^{-2Nt^2}. \tag{2.16}$$

In [51], a fixed sample size test was proposed based on (2.16) (the authors of [51] call the method based on this test '*approximate model checking*'). The investigator chooses a significance parameter α and a so-called '*approximation*' parameter ϵ. She then draws

$$N = \frac{\log\left(\frac{2}{\alpha}\right)}{2\epsilon^2} \tag{2.17}$$

samples. We can then rewrite (2.16) to

$$\mathbb{P}(|\bar{X} - \pi_0| \geq \epsilon) \leq \alpha. \tag{2.18}$$

The test is then as follows: we draw N samples and check if $|\bar{X} - \pi_0| > \epsilon$. If so, we reject the null hypothesis, otherwise the test is inconclusive. If we reject the null hypothesis, we accept H_{+1} if $\bar{X} > \pi_0$ and we accept H_{-1} otherwise. The test satisfies (2.3) and (2.4) because under the null hypothesis $\mathbb{E}(\bar{X}) = \pi_0$, so that (2.18) is really an upper bound for the probability of rejecting the null hypothesis when it is valid.

The interpretation of ϵ is not immediately clear. The choice of ϵ does not affect the correctness, as the requirements on the probability of error of the first type always hold, also when $|\pi - \pi_0| < \delta$. This means that ϵ is not comparable to the indifference level δ of the SPRT. However, ϵ does have an impact on the power, so we can use it to establish an upper bound on the error probability of the second type. Assume, without loss of generality, that H_{+1} holds, so that $\pi = \pi_0 + \gamma$ for some $\gamma > 0$. Note that we can use (2.17) to write ϵ as a function of N. For an error of the second type to occur it must hold that after N samples we have that $\bar{X} - \pi_0 < \epsilon$. We can use (2.15) and the fact that $\mathbb{E}(\pi_0 - \bar{X}) = -\gamma$ to establish

$$\begin{aligned}
\mathbb{P}(\bar{X} - \pi_0 < \epsilon) &= \mathbb{P}(\pi_0 - \bar{X} > -\epsilon) \\
&= \mathbb{P}(\pi_0 - \bar{X} + \gamma > \gamma - \epsilon) \\
&\leq e^{-2N(\gamma-\epsilon)^2}.
\end{aligned}$$

Setting $\beta_1 = e^{-2N(\gamma-\epsilon)^2}$ means that (2.5) is valid. It has two solutions for N, one of which gives positive $\gamma - \epsilon_N$ (which is a requirement of the Chernoff-Hoeffding

bound). This solution is given by

$$N = \frac{2\sqrt{\log(\beta_1)\log(\frac{\alpha}{2})} - \log(\frac{\alpha\beta_1}{2})}{2\gamma^2}. \tag{2.19}$$

A similar argument can be made for $\gamma < 0$, β_2 and (2.6). Since we choose $\beta_1 = \beta_2$, we obtain the same value for N for both error probabilities of the second type. An interesting question concerns the relationship between this value for N and a

		$\pi_0 = 0.5$		$\pi_0 = 0.2$	
α	γ	N_C	N_G	N_C	N_G
0.05	0.1	667	319	667	164
	0.05	2667	1294	2667	753
	0.01	66666	32481	66666	20428
0.025	0.1	806	434	806	221
	0.05	3223	1757	3223	1020
	0.01	80560	44121	80560	27738

Table 2.1: Chernoff (N_C) and Gaussian (N_G) sample sizes, $\beta = \alpha$.

value N' that would be the result of a Gaussian estimate for the distribution of S_N, found by setting (2.11) equal to α. The actual difference is shown for several different choices of α and ϵ in Table 2.1. One thing to note is that the Chernoff test's sample size does not depend on π_0, while the Gauss test's sample size does. Another thing is that the sample size for the Chernoff test always seems to be larger than for the Gauss test. This is the price of using an exact result instead of an asymptotic approximation.

To summarise this section and Section 2.2.1: we have discussed three possible ways to come up with a choice for N for a fixed sample size test: using the exact binomial distribution of S_N, using a Gaussian approximation or using the Chernoff-Hoeffding bound. The time complexity of the binomial approach increases linearly in N, the Gaussian sample sizes rely on an approximation that is bad for small N and the Chernoff-based bound does not depend on the value of π_0 (and has a mild increase in sample size). Hence, the optimal choice depends on the application and the investigator's preferences.

2.2.4 Bayesian SPRT

In [57] an approach based on Bayesian likelihood ratios was proposed. In Bayesian statistics, the true parameter π is itself seen as the realisation of a random variable. To implement the method, a *prior distribution* $\mathbb{G}$ must be given on $[0, 1]$ such that the probability that the true parameter π is some set $A \subset [0, 1]$ is assumed to be given by $\int_A d\mathbb{G}$. Let $\vec{X} = (X_1, \ldots, X_N)$ be a random variable with $X_i \in \mathcal{X}$, $i = 1, \ldots, N$. If the probability function $\mathbb{P}(\vec{X} = \vec{x})$ of $\vec{X}$ is given by a function $f_{\vec{X},\pi}(\vec{x})$ that depends

on a parameter π, then the $\mathbb{G}$-weighted likelihood is given by

$$\mathbb{F}_{\mathbb{G}}(\vec{X} = \vec{x}|A) = \int_A f_{\vec{X},\pi}(\vec{x}) d\mathbb{G}(\pi).$$

The prior $\mathbb{G}$ is usually chosen to be a Beta distribution, as it is the so-called *conjugate prior* of the binomial distribution (that is, if X has a binomial distribution then the posterior distribution is again a Beta distribution, which is computationally attractive). Assume that we have drawn a sample $x_1, \ldots, x_N$ of realisations of (2.1) from the set $\mathcal{X}^N$ of all possible samples, with N yet to be determined. With $s_N = \sum_{i=1}^N x_i$, define $f_{\vec{X},\pi}(\vec{x}) = \pi^{s_N}(1-\pi)^{s_N}$, so that $f_{\vec{X},\pi}(\vec{x})$ is the product of the probabilities of the individual realisations $x_1, \ldots, x_N$. The test statistic is the *Bayes factor*, given by

$$B(\vec{x}) = \frac{\mathbb{F}_{\mathbb{G}}(\vec{X} = \vec{x}|H_{+1})}{\mathbb{F}_{\mathbb{G}}(\vec{X} = \vec{x}|H_{-1})}.$$

Assume that we perform a sequential test in which we draw samples from (2.1) until we can accept H_{+1} or H_{-1}; we accept H_{+1} if after N samples $B(\vec{X}) > T$ and accept H_{-1} if after N samples $B(\vec{X}) < \frac{1}{T}$, where the N-th sample is the first sample for which either condition holds. *Under the prior*, the error probability is then bounded from above by $1/T$. As in [119], one proves (with A_{-1} the event of accepting H_{-1}):

$$\begin{aligned}
\mathbb{F}_{\mathbb{G}}(A_{-1}|H_{+1}) &= \sum_{\vec{x}\in\mathcal{X}^N} \mathbf{1}\left(B(\vec{X}) < \frac{1}{T}\right) \mathbb{F}_{\mathbb{G}}(\vec{X} = \vec{x}|H_{+1}) \\
&< \sum_{\vec{x}\in\mathcal{X}^N} \mathbf{1}\left(B(\vec{X}) < \frac{1}{T}\right) \frac{\mathbb{F}_{\mathbb{G}}(\vec{X} = \vec{x}|H_{-1})}{T} \\
&\leq \frac{1}{T} \sum_{\vec{x}\in\mathcal{X}^N} \mathbb{F}_{\mathbb{G}}(\vec{X} = \vec{x}|H_{-1}) = \frac{1}{T}.
\end{aligned}$$

If the prior $\mathbb{G}$ is defined by a Beta distribution function with shape parameters a and b, written as $F_{a,b}$, then the Bayes factor at step N can be rewritten for computational reasons to

$$B(\vec{X}) = \frac{1}{F_{a+S_N, N-S_N+b}(\pi_0)} - 1$$

The edges of the critical areas as a function of N can be found by inverting $B(\vec{X})$ with respect to S_N. In Section 2.4.1, we will demonstrate through numerical examples that the critical area boundaries seem to behave similar to $\sqrt{N}$ (we are not aware of theoretical results on the subject). The exact width of the boundaries depends on the prior $\mathbb{G}$, characterised by the two shape parameters a and b of the Beta distribution. In the case $a = b = 1$, the Beta distribution equals the uniform distribution on $[0, 1]$, and the prior is said to be *uniform* — the uniform prior is often used in practice (e.g., in the case studies of [57]) as a default.

We note here that the dependence on the prior is substantial. A prior with shape parameters a and b can be seen as starting the testing procedure with a uniform prior and $a + b - 2$ samples already drawn, with $a - 1$ times a 1 being drawn and $b - 1$ times a 0. It is clear that this has a major impact on the correctness of the test: for each π_0, values a and b can be chosen such that a conclusion will be drawn after a single sample, *regardless of the value of the drawn sample*. Moreover, the uniform prior is in itself a strong assumption, and despite it being nicknamed as the '*non-informative prior*' it should not be used to reflect a total lack of information about the true probability π. In fact, if π_0 is close to 0 or 1, one can again reach the situation in which a conclusion is drawn after the first sample, irrespective of its value. More on that in Section 2.4.2.

2.3 The New Supergaussian Sequential Tests

In the previous section, we have discussed four statistical tests that all require model parameters that determine whether a correct and satisfying result is returned. For the two fixed sample tests (Gauss and Chernoff), this parameter is the fixed sample size N. If chosen too small, the test terminates inconclusively, leaving the investigator without an answer. For the two sequential tests — SPRT and Bayes — an indifference level δ and a prior $\mathbb{G}$ have to be chosen respectively, and if chosen incorrectly, an answer would be returned that does not satisfy the requirements on the probability of error, without giving any hint that this problem occurred. In this section we will introduce two sequential tests that do not have this problem, and which depend only on model parameters in terms of their efficiency.

The outline of this section is as follows. In Section 2.3.1 we will formulate the problem in terms of the shape of $\mathcal{NC}$. In Section 2.3.2 we will introduce a new test based on original analysis, and in Section 2.3.3 we will introduce a test which is an application of the so-called power one tests proposed in [28]. The dependence of the efficiency of the new tests on their model parameters will be investigated in Section 2.3.4. The performance of the new tests will be evaluated empirically in Section 2.4.

2.3.1 The Desired Shape of the Non-Critical (NC) Areas

As was explained in Section 2.1, the principle of statistical model checking is to use discrete-event simulation to draw sample paths in order to produce a test statistic Z_N that satisfies

$$Z_N - Z_{N-1} = \begin{cases} 1 - \pi_0 & \text{with probability } \pi, \\ -\pi_0 & \text{with probability } 1 - \pi. \end{cases}$$

Since the focus of this section will be on sequential tests, our tests are characterised by areas $\mathcal{U}$ and $\mathcal{L}$ such that if Z_N enters these areas, we accept H_{+1} and H_{-1} respectively (see Figure 2.2b). These areas are in turn completely determined by the

boundaries $u(n)$ and $l(n)$ between $\mathcal{U}$ and $\mathcal{L}$ respectively and $\mathcal{NC}$. We assume that the tests are symmetric (i.e., $u(n) = -l(n)$), which means that we are looking for a function $u(n)$ such that (2.3-2.6) are satisfied. In the following we will discuss what general shape this function must have in order to satisfy these properties without requiring an indifference region or a valid guess.

First note that $\mathbb{E}Z_N = N(\pi - \pi_0)$, so that under H_0 the process Z_N is expected to stay around zero while under the alternative hypotheses Z_N is expected to drift away linearly from 0. For the SPRT, none of (2.3–2.6) will hold for general $\pi - \pi_0$. However, they do hold for all $\pi - \pi_0 \notin [-\delta, \delta]$, and the assumptions of the test mean that $\pi - \pi_0$ must satisfy this condition anyway.

In the following, we will argue that if l and u diverge slower than linearly but faster than $\sqrt{N}$, we can establish bounds for general $\pi - \pi_0$. For a test in which l and u diverge linearly — that is, for fixed constants $c_i > 0$, $i \in \{1, \ldots, 4\}$, set $l(N) = -c_1 N - c_2$ and $u(N) = c_3 N + c_4$ — it holds that for each α_i constants c_i can be picked such that (2.3) and (2.4) hold for all $\pi - \pi_0$ (because of Proposition 6.5.1 of [99]). However, (2.5) and (2.6) will not hold for arbitrarily small $\pi - \pi_0$.

The boundaries of the Gauss test have the form $u(N) = a\sqrt{N+k},\ \ a, k > 0$. From the literature on Wiener processes [109], we know that these boundaries are crossed after a finite amount of time with probability 1, and that the expected time until the boundary is crossed is finite if $a < 1$ and infinite otherwise, regardless of k. So for the Gauss test, (2.3) and (2.4) will also not hold. From this we can conclude that the size of $\mathcal{NC}$ needs to grow asymptotically faster than $\sqrt{N}$ and slower than linearly.

In the following, we investigate two distinct types of $\mathcal{NC}$-area upper bounds that grow asymptotically faster than $\sqrt{N}$ and slower than linearly, namely bounds of the form $a(N+k)^b$, with $b \in (\frac{1}{2}, 1)$, and bounds of the form $a\sqrt{(N+k)\log(N+k)}$. We will introduce two different tests based on these types of bounds in Sections 2.3.2 and 2.3.3 respectively, both using different techniques to ensure the bounds (2.3) and (2.4).

2.3.2 Azuma Test

The test of this section is based on Proposition 6.5.1 of [99]. We begin with some background on proving the correctness of a test. In order to prove correctness, the fundamental difference between a fixed sample size test and a sequential test is that one needs to bound the probability under H_0 that $Z_N \in [-u(N), u(N)]$ for a value N which is 1) known a priori for the former and 2) a random variable for the latter. For N fixed, this probability of interest is a tail probability, and several well-known bounds from probability theory deal with bounds on or approximations of tail probabilities.

The Gauss test of Section 2.2.1 uses the *Central Limit Theorem*, which gives an approximation for the tail probability of sums of independent, identically distributed random variables. The Chernoff test of Section 2.2.3 used the *Chernoff-Hoeffding bound*, which establishes an exact (but looser) bound on tail probabilities of sums

of i.i.d. random variables. Another bound that can be applied in our setting is the *Azuma-Hoeffding inequality*, which gives a bound on the tail probabilities of *martingales with bounded differences*.[15] Since our process Z_N is a martingale with bounded differences, the Azuma-Hoeffding Inequality could be used to establish a fixed sample size test, even if the Chernoff-Hoeffding bound may be tighter. The possibility of such a test was mentioned in [66], and has recently been investigated in [104] and [105].

While the Azuma-Hoeffding theorem establishes a bound for the probability that Z_N is outside $[-u(N), u(N)]$, Ross shows in Section 6.5 of [99] that a bound can also be established for the probability that Z_M is outside $[-u(M), u(M)]$ for *any* $M \geq N$, and showed that these bounds were *equal*. He calls this theorem "*The Generalized Azuma Inequality*". The proof of the theorem is a simple application of Proposition 6.5.1 of [99], which determines an upper bound for the probability that a martingale with bounded differences ever crosses some line $aN + b$, with $a, b > 0$.

In this section we will establish a similar bound for the probability that a martingale with bounded differences eventually leaves an area $[-u(N), u(N)]$, where $u(N) = a(N + k)^b$, $b \in [\frac{1}{2}, 1)$ — see Theorem 2.1. The proof is similar, but requires an additional lemma, and we need the fact that our region has both a lower and upper boundary for our proof technique to work. The basis of our upper bound is that, under the null hypothesis, the process Z_n is a *martingale*, i.e., a stochastic process for which it holds that $\mathbb{E}(Z_n|Z_{n-1}, \ldots, Z_0) = Z_{n-1}$, and that the size of each step X_i is bounded. This allows us to use a similar technique as the one used to prove Proposition 6.5.1 of [99]. We need the following two lemmas from [99]:

Lemma 2.1. *Let X be such that $\mathbb{E}(X) = 0$ and $\mathbb{P}(-a \leq X \leq b) = 1$. Then for each convex function f it holds that*

$$\mathbb{E}(f(X)) \leq \frac{b}{a+b} f(-a) + \frac{a}{a+b} f(b).$$

Proof. See Lemma 6.3.1 (page 305) of [99]. □

Lemma 2.2. *For $\theta \in [0, 1]$*

$$\theta e^{(1-\theta)x} + (1-\theta)e^{-\theta x} \leq e^{x^2/8}.$$

Proof. See Lemma 6.3.2 (page 306) of [99]. □

We also need the following two new lemmas:

[15] A martingale is a stochastic process Y_N for which it holds that $\mathbb{E}(Y_N|Y_{N-1}, \ldots, Y_0) = Y_{N-1}$, and if $\mathbb{P}(|Y_N - Y_{N-1}| < c) = 1$ for some constant $c < \infty$ and all $N \geq 0$ then the martingale is said to have bounded differences.

Lemma 2.3. *For $f_n = a(n+k)^b$, $f'_n = \frac{d}{dn} f_n$, it holds for $n \in \mathbb{R}, n+k > 0$, $a > 0$ and $b \in (\frac{2}{3}, 1)$ that*

$$f_{n-1} - f_n + \left(3 - \frac{2}{b}\right) f'_n + 2\left(\frac{f'_{n-1}}{f'_n} - 1\right) f_{n-1} \leq 0 \tag{2.20}$$

Proof. Writing $z \triangleq \frac{n+k-1}{n+k}$,

$$g_b(z) \triangleq 2z^{2b-1} - z^b - 1 + (3b-2)(1-z),$$

$g_b(z)$ equals the left-hand side of (2.20) after dividing by f_n. Since the statements $n + k = \frac{1}{1-z} \in (0, \infty)$ and $z \in (0, 1)$ are equivalent, we only need to evaluate whether g_b is negative on $(0, 1)$. Since $g_b(1) = 0$, we are done if we can show that $\frac{d}{dz} g_b(z)$ is positive on $(0, 1)$. Writing $y \triangleq z^{b-1}$, we have that

$$\frac{d}{dy} g_b(y) = (4b-2)\, y^2 - by - (3b-2)$$

We find that this parabola has two roots, namely 1 and $\frac{2-3b}{4b-2}$. The second root is negative if $b \in (\frac{2}{3}, 1)$, which proves the lemma. □

Lemma 2.4. *Let X_i, $i \in \mathbb{N}$, be i.i.d. such that $\mathbb{P}(0 \leq X_i \leq 1) = 1$ and $\mathbb{E}(X_i) = \pi_0$. Let $S_n = \sum_{i=1}^n X_i$ and $Z_n = S_n - n\pi_0$. Let $f_n = a(n+k)^b$ with $k > 0$, $a > 0$ and $b \in (\frac{2}{3}, 1)$, and let the process be stopped at $-f_n$ and f_n (i.e., if $\exists m$ s.t. $Z_m < -f_m$ then $Z_m := -f_m$ and $\forall m' > m$ $Z_{m'} = -f_m$ — similarly if $\exists m$ s.t. $Z_m > f_m$). Let $c_n = 8(3 - \frac{2}{b})\frac{d}{dn} f_n$. Then*

$$W_n \triangleq e^{c_n(Z_n - f_n)}. \tag{2.21}$$

is a supermartingale, i.e., $\mathbb{E}(W_n | W_{n-1}, \ldots, W_1) \leq W_{n-1}$.

Proof. The process W_n is a supermartingale if and only if

$$\frac{\mathbb{E}(W_n | \mathcal{F}_{n-1})}{W_{n-1}} \leq 1$$

with $\mathcal{F}_n \triangleq W_n, \ldots, W_1$. Now,

$$\begin{aligned} \frac{W_n}{W_{n-1}} &= \frac{e^{c_n(Z_n - f_n)}}{e^{c_{n-1}(Z_{n-1} - f_{n-1})}} \\ &= \frac{e^{c_{n-1} f_{n-1}}}{e^{c_n f_n}} \cdot \frac{e^{c_n Z_n}}{e^{c_{n-1} Z_{n-1}}} \end{aligned}$$

Note that,

$$\begin{aligned} \mathbb{E}\left(\frac{e^{c_n Z_n}}{e^{c_{n-1} Z_{n-1}}} | \mathcal{F}_{n-1}\right) &= \mathbb{E}(e^{c_n \cdot (X_n - \pi_0)}) \cdot \mathbb{E}(\frac{e^{c_n Z_{n-1}}}{e^{c_{n-1} Z_{n-1}}} | \mathcal{F}_{n-1}) \\ &= \mathbb{E}(e^{c_n \cdot (X_n - \pi_0)}) \cdot W_{n-1}^{\frac{c_n - c_{n-1}}{c_{n-1}}} \cdot \frac{e^{c_n f_{n-1}}}{e^{c_{n-1} f_{n-1}}} \end{aligned}$$

which means that

$$\mathbb{E}(W_n|\mathcal{F}_{n-1}) = W_{n-1}^{\frac{c_n}{c_{n-1}}} \cdot \frac{e^{c_n f_{n-1}}}{e^{c_n f_n}} \cdot \mathbb{E}(e^{c_n \cdot (X_n - \pi_0)})$$

So for W_n to be a supermartingale, it must hold that

$$\frac{e^{c_n f_{n-1}}}{e^{c_n f_n}} \cdot \mathbb{E}(e^{c_n \cdot (X_n - \pi_0)}) \leq W_{n-1}^{1-\frac{c_n}{c_{n-1}}}.$$

From Lemmas 2.1 and 2.2 we know that this is implied by

$$e^{c_n f_{n-1} - c_n f_n} \cdot e^{\frac{1}{8}c_n^2} \leq W_{n-1}^{1-\frac{c_n}{c_{n-1}}}$$

or

$$c_n f_{n-1} - c_n f_n + \tfrac{1}{8}c_n^2 \leq \log W_{n-1}^{1-\frac{c_n}{c_{n-1}}}.$$

We also know that

$$\begin{aligned} \log W_{n-1}^{1-\frac{c_n}{c_{n-1}}} &= \log e^{(c_{n-1}-c_n)\cdot(Z_{n-1}-f_{n-1})} \\ &= (c_{n-1} - c_n) \cdot (Z_{n-1} - f_{n-1}). \end{aligned}$$

We know that c_n is positive and decreasing in n, and by implication that $c_{n-1} - c_n > 0$. This means that for the inequality to hold we must find the lowest possible Z_{n-1}. Normally, Z_{n-1} can be as low as $-(n-1)p$. However, because the process is stopped also at $-f_{n-1}$ this is the lowest possible value. Hence, we only have to show whether

$$f_{n-1} - f_n + \frac{1}{8}c_n \leq -2\left(\frac{c_{n-1}}{c_n} - 1\right) f_{n-1}, \tag{2.22}$$

and this follows from Lemma 2.3. □

We can then prove the following theorem:

Theorem 2.1. *Again, let X_i, $i \in \mathbb{N}$, be i.i.d. such that $\mathbb{P}(0 \leq X_i \leq 1) = 1$ and $\mathbb{E}(X_i) = \pi_0$, and let $S_n = \sum_{i=1}^n X_i$ and $Z_n = S_n - n\pi_0$. Let $f_n = a(n+k)^b$ with $k > 0$, $a > 0$ and $b \in (\frac{2}{3}, 1)$. Then,*

$$\mathbb{P}(\exists n \geq 0 : Z_n > f_n) \leq e^{-8(3b-2)a^2k^{2b-1}} \tag{2.23}$$

Proof. Let $W_n = e^{c_n(Z_n - f_n)}$. Since we know that W_n is a supermartingale by Lemma 2.4, we can define the bounded stopping time

$$N(m) = \min\{n : |Z_n| \geq f_n \text{ or } n = m\}$$

to find that

$$\begin{aligned}\mathbb{P}(Z_{N(m)} \geq f_{N(m)}) = &\quad \mathbb{P}(W_{N(m)} \geq 1) \\ \leq &\quad \mathbb{E}(W_{N(m)}) \\ \leq &\quad \mathbb{E}(W_0) = e^{-f(0)c(0)} \\ = &\quad e^{-8(3-\frac{2}{b})ba^2k^{2b-1}},\end{aligned}$$

where the first inequality holds because of the Markov inequality, and the second inequality because of the Martingale Stopping Theorem (Theorem 6.2.2 of [99]) and the fact that $N(m)$ is bounded. Theorem 2.1 then follows from taking the limit of $m \to \infty$. □

The theorem gives us the following corollary:

Corollary 2.1. *If we carry out the test of Section 2.1 with the boundary between $\mathcal{U}$ and $\mathcal{NC}$ defined by $u(N) = a(N+k)^b$ and the boundary between $\mathcal{L}$ and $\mathcal{NC}$ defined by $-u(N)$. Then* (2.3), (2.4), (2.5) *and* (2.6) *are all satisfied, with*

$$\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = e^{-8(3b-2)a^2k^{2b-1}}$$

Proof. Follows directly from Theorem 2.1. □

Only one of the following claims is true: H_{+1}, H_{-1} or H_0. If the first or second of these claims is true, the error probability is β_1 or β_2. If the third claim is true, the probability of error is $\alpha_1 + \alpha_2$. As we argued in Section 2.1.2, we assume H_0 not to hold. Still, to be on the safe side we will focus on the worst-case behaviour and fix $\alpha = 2 \cdot e^{-8(3b-2)a^2k^{2b-1}}$ whenever we apply the test of Corollary 2.1.

Throughout this thesis we will call the test of Corollary 2.1 the '*Azuma*' test. In Section 2.3.4 we will further elaborate on how to choose the parameters a and k.

2.3.3 Darling Test

In Section 2.3.1 we argued why the boundaries of $\mathcal{NC}$ need to grow faster than $\sqrt{N}$ for (2.3) and (2.4) to hold. Not only bounds of the form $a(n+k)^b$ satisfy this property; we can establish similar results for $a\sqrt{(N+k)\log(N+k)}$, but we will start with a discussion on how to arrive at these results. In [27], it was proven that for $\mathcal{NC}$ widths of order greater than $\sqrt{cN \log\log N}$, for some $c > 2$, the probability that it will be left is smaller than 1. Based on [27], it was proven in [28] (Theorem 3) that for the test of Section 2.1 for general $\mathcal{U}$-$\mathcal{NC}$ boundary $u(N)$ and $\mathcal{L}$-$\mathcal{NC}$ boundary $-u(N)$, if one could find an $\epsilon > 0$ such that

$$\sum_{n=1}^{\infty} e^{-\frac{u^2(n)}{n+1}} \leq \epsilon \tag{2.24}$$

then the probability of error was bounded from above by $2\sqrt{2}\epsilon$. The idea is then to carry out the test of Section 2.1, with $u(N)$ chosen such that (2.24) can be used to

show that (2.3-2.6) hold. The question is then how to choose $u(N)$, and whether the bound of (2.24) is better than the bound of Theorem 2.1 for the choice of $u(N)$.

For boundaries of the form $u(N) = a(N+k)^b$, the bound of (2.24) can be much weaker than the bound of Theorem 2.1. For example, for $u(N) = (N+2)^{3/4}$, the bound of Theorem 2.1 evaluates to about 0.1182 while using (2.24) we cannot obtain a bound that is much better than 1.8821, so there is not much merit in using (2.24) for boundaries of this form. On the other hand, our proof of Theorem 2.1 requires analytical steps that do not work for boundaries that are of order $N^{2/3}$ or tighter. For boundaries of the form $u(N) = \sqrt{a(N+1)\log(N+k)}$ the summation on the left-hand side of (2.24) equals the Hurwitz zeta function evaluated in a and k. This means that we now have a choice between a relatively strong bound for the case $N^b, b \in (\frac{2}{3}, 1)$, and a relatively weak bound for the case $\sqrt{N \log N}$. For boundaries of the latter type, a rather large value of k must be chosen in order to obtain bounds for reasonable values of α, but asymptotically the non-critical area will become much smaller than if an order N^b boundary is chosen. This has an impact on the efficiency of the two methods, as we will see in Section 2.3.4.

Throughout the rest of this thesis we will apply (2.24) only to boundaries of the form $u(N) = \sqrt{a(N+1)\log(N+k)}$. We will call the test of Section 2.1 carried out using this boundary the '*Darling*' test.

2.3.4 Optimal Parameter Choice

To summarise, we now have two new tests, with non-critical area upper boundaries $u(N; a, k)$ given by $a(N+k)^b$ and $\sqrt{a(N+1)\log(N+k)}$ respectively. Both depend on parameters a and k that in both settings have the same influence, while the Azuma test also depends on a third parameter b. The parameter a influences the increase in width of $\mathcal{NC}$ and its influence does not fade relative to N when N grows large. A high value of parameter k shifts the cone that defines $\mathcal{NC}$ (see Figure 2.2b) to the left, making the area $\mathcal{NC}$ wider for small values of N) — it can be seen as a startup parameter in some sense: if two tests are carried out with the same a and $k_1 < k_2$ respectively, the second test can be seen as the first test with Z_N still at zero after $N = k_2 - k_1$ samples have been drawn. A high value for b means that the area $\mathcal{NC}$ boundary $u(N)$ will more closely resemble a straight line, which means that it will grow much wider asymptotically.

A high value for k makes it harder to accept an alternative hypothesis in the beginning, but — since a and b can be chosen smaller to maintain the same significance level α — easier to reject as N grows bigger. Since the upper bound on the probability of error is fixed to equal α, k can be determined as a function of a, α and b. For the Azuma test, we easily derive from Corollary 2.1 that

$$k_{\text{Azuma}}(a, \alpha, b) = \left(\frac{\log\left(\frac{\alpha}{2}\right)}{8a^2(2-3b)} \right)^{\frac{1}{2b-1}}$$

For the Darling test, it is harder to obtain a similar expression from (2.24) since we have to solve for the lower bound of a summation, but for practical purposes the

summation in (2.24) can be approximated by the integral

$$\int_1^\infty e^{-\frac{u^2(x)}{x+1}} dx.$$

We then derive

$$k_{\text{Darling}}(a, \alpha) = \left(\frac{\alpha(a-1)}{2\sqrt{2}} \right)^{-\frac{1}{a-1}} - 1.$$

We then want to minimise the expected number of samples drawn, which we approximate using the intersection of the expected trajectory of Z_N — which equals $|\pi - \pi_0|N$ — and $u(N)$. This means that we have to solve

$$|\pi - \pi_0|N = u(N; a, k(a, \alpha)) \tag{2.25}$$

for N and then minimise over a. Unfortunately, both in the case of the Azuma and the Darling test, solving (2.25) for N does not lead to a closed form expression. However, in both cases we can do the minimisation numerically, since the function $u(N) - |\pi - \pi_0|N$ has a derivative simple enough to allow for Newton's method to find its roots. We seek the minimum of $N(a)$ for $a \in [0, \infty)$, but for the sake of being able to use straightforward numerical techniques, we search for the minimum of $N(\frac{1}{1+a})$ for $\frac{1}{1+a} \in (0, 1]$. Since this is a bounded interval, we can use techniques such as golden section search [20] to find the minimum. For the Darling test we even know that $a > 1$, meaning that we can minimise $N(\frac{1}{a})$ on $(0, 1]$.

In Table 2.2, we show the (approximately) optimal parameters a and k that we found for both tests for several values of γ (recall that this is our guess for $|\pi - \pi_0|$). We can see that for the Azuma test, a grows proportional to $\sqrt{\gamma}$, and k inversely proportional to γ^2.

The final remaining value to choose is then the parameter b of the Azuma test. A higher value for b means a tighter bound on the error probability of the first type, but the area $\mathcal{NC}$ will grow larger asymptotically. The difference in terms of the tightness of the bound can be observed in Table 2.3, where we display the solutions to equation (2.25) for the Azuma test with several values of b and the Darling test (with a and k chosen optimally). The impact of a low value for b is twofold: the

γ	Azuma a	Azuma k	Darling a	Darling k
10^{-1}	0.3452	$2.40{\cdot}10^{2}$	1.6686	$7.63{\cdot}10^{2}$
10^{-2}	0.1092	$2.40{\cdot}10^{4}$	1.4363	$6.97{\cdot}10^{4}$
10^{-3}	0.0345	$2.40{\cdot}10^{6}$	1.3278	$6.68{\cdot}10^{6}$
10^{-4}	0.0110	$2.40{\cdot}10^{8}$	1.2644	$6.54{\cdot}10^{8}$
10^{-5}	0.0035	$2.40{\cdot}10^{10}$	1.2225	$6.46{\cdot}10^{10}$
10^{-6}	0.0011	$2.40{\cdot}10^{12}$	1.1927	$6.42{\cdot}10^{12}$

Table 2.2: Approximately optimal parameter choices for $\alpha = 0.05$. For this table we used $b = \frac{3}{4}$

expected number of needed samples when the guess is correct will be higher, but the test will become less sensitive to the guess γ. Note, however, that even for very low values of b (e.g., 0.67), the Azuma test will still be more sensitive than the Darling test. Since for $b = 0.67$ the Azuma test has a higher expected number of needed samples than the Darling test, while it is still less sensitive, the Azuma test has no advantages over the Darling test so we can say that it performs strictly worse than the Darling test. The choice $b = 0.99$ on the other hand leads to enormous parameter sensitivity. Values of b around $\frac{3}{4}$ seem to strike a nice balance, and in Section 2.4, where we empirically validate the analysis of this section, we will only consider the Azuma test with this parameter choice.

2.4 Results and Comparisons

In this section we compare the performance of the existing tests discussed in Section 2.2 to the new tests introduced in Section 2.3 — see Table 2.5 for a summary. We do this in two ways: we will begin in Section 2.4.1 by comparing the tests in terms of the implied test decision areas as discussed in Section 2.1, and see how these ar-

$\lvert\pi - \pi_0\rvert$	γ	Azuma			Darling
		$b = 0.67$	$b = 0.75$	$b = 0.9$	
10^{-1}	10^{-1}	$\mathbf{9.85{\cdot}10^{3}}$	$\mathbf{4.792{\cdot}10^{2}}$	$\mathbf{1.875{\cdot}10^{2}}$	$\mathbf{1.272{\cdot}10^{3}}$
	10^{-2}	$4.258{\cdot}10^{4}$	$2.249{\cdot}10^{3}$	$9.842{\cdot}10^{2}$	$1.606{\cdot}10^{3}$
	10^{-3}	$3.961{\cdot}10^{5}$	$2.116{\cdot}10^{4}$	$9.394{\cdot}10^{3}$	$2.088{\cdot}10^{3}$
	10^{-4}	$3.933{\cdot}10^{6}$	$2.104{\cdot}10^{5}$	$9.351{\cdot}10^{4}$	$2.567{\cdot}10^{3}$
10^{-2}	10^{-1}	$5.837{\cdot}10^{6}$	$1.421{\cdot}10^{6}$	$7.998{\cdot}10^{72}$	$2.041{\cdot}10^{5}$
	10^{-2}	$\mathbf{9.85{\cdot}10^{5}}$	$\mathbf{4.792{\cdot}10^{4}}$	$\mathbf{1.875{\cdot}10^{4}}$	$\mathbf{1.784{\cdot}10^{5}}$
	10^{-3}	$4.258{\cdot}10^{6}$	$2.249{\cdot}10^{5}$	$9.842{\cdot}10^{4}$	$2.091{\cdot}10^{5}$
	10^{-4}	$3.961{\cdot}10^{7}$	$2.116{\cdot}10^{6}$	$9.394{\cdot}10^{5}$	$2.567{\cdot}10^{5}$
10^{-3}	10^{-1}	$6.251{\cdot}10^{9}$	$1.42{\cdot}10^{10}$	$7.998{\cdot}10^{172}$	$2.865{\cdot}10^{7}$
	10^{-2}	$5.837{\cdot}10^{8}$	$1.421{\cdot}10^{8}$	$7.993{\cdot}10^{74}$	$2.444{\cdot}10^{7}$
	10^{-3}	$\mathbf{9.85{\cdot}10^{7}}$	$\mathbf{4.792{\cdot}10^{6}}$	$\mathbf{1.875{\cdot}10^{6}}$	$\mathbf{2.284{\cdot}10^{7}}$
	10^{-4}	$4.258{\cdot}10^{8}$	$2.249{\cdot}10^{7}$	$9.842{\cdot}10^{6}$	$2.571{\cdot}10^{7}$
10^{-4}	10^{-1}	$6.703{\cdot}10^{12}$	$1.42{\cdot}10^{14}$	$7.998{\cdot}10^{272}$	$3.675{\cdot}10^{9}$
	10^{-2}	$6.251{\cdot}10^{11}$	$1.42{\cdot}10^{12}$	$7.993{\cdot}10^{174}$	$3.141{\cdot}10^{9}$
	10^{-3}	$5.837{\cdot}10^{10}$	$1.421{\cdot}10^{10}$	$7.995{\cdot}10^{76}$	$2.893{\cdot}10^{9}$
	10^{-4}	$\mathbf{9.85{\cdot}10^{9}}$	$\mathbf{4.792{\cdot}10^{8}}$	$\mathbf{1.875{\cdot}10^{8}}$	$\mathbf{2.776{\cdot}10^{9}}$

Table 2.3: For each combination $(\gamma, |\pi - \pi_0|, \text{test type})$, we display the solution to (2.25) — i.e., the N for which the expected trajectory leaves $\mathcal{NC}$ — with parameters a and k chosen optimally. Bold values imply that $\gamma = |\pi - \pi_0|$, i.e., that the guess is correct.

eas behave as a function of the number of samples drawn. In Section 2.4.2, we will then compare the tests by the three performance measures mentioned in the introduction: the correctness, the power and the efficiency. These measures may depend on the guess γ, the indifference level δ or the prior measure $\mathbb{G}$, as summarised in Table 2.4.

Test	Correctness	Power	Efficiency
Gauss	—	γ	γ
SPRT	δ	N.A.	δ
Chernoff	—	γ	γ
Bayes	$\mathbb{G}$	—	$\mathbb{G}$
Azuma	—	—	γ
Darling	—	—	γ

Table 2.4: Parameter dependence.

2.4.1 Shape of the Non-Critical Areas (NC)

As explained in Section 2.1, all of the tests in this chapter can be considered in the context of a single framework: a random walk Z_n that always jumps up by $1-\pi_0$ with probability π or down by π_0 with probability $1-\pi$. The tests can then be defined in terms of the boundaries of the test decision areas (see Figure 2.2). The most fundamental property of the test decision areas is the shape of $\mathcal{NC}$, displayed for each test in Table 2.5. In this section we will zoom in on these shapes, particularly

Test	Type	Source	Section	NC Area Width
Gauss	fixed N	CLT	2.2.1	$O(\sqrt{N})$
SPRT	sequential	[111], [115]	2.2.2	constant
Chernoff	fixed N	[51]	2.2.3	$O(\sqrt{N})$
Bayes	sequential	[54], [57], [119]	2.2.4	$O(\sqrt{N})$
Azuma	sequential	this thesis	2.3.2	$O(N^b), b \in (\frac{2}{3}, 1)$
Darling	sequential	[28], this thesis	2.3.3	$O(\sqrt{N \log N})$

Table 2.5: Summary of the tests.

on how the tests compare to each other in this respect. Note that the areas $\mathcal{NC}$ of all the old tests except the SPRT widen approximately as$\sqrt{N}$, while the areas $\mathcal{NC}$ of the new tests widen faster.

In Figure 2.4, we compare the non-critical areas $\mathcal{NC}$ for the Gauss, SPRT, Azuma and Darling test for the symmetrical situtation $\pi_0 = \frac{1}{2}$. The area $\mathcal{NC}$ is narrower for the Azuma test than for the Darling test for small values of N, but the Azuma boundaries eventually overtake those of the Darling test. This is obvious as functions of the type $N^{\frac{3}{4}}$ are asymptotically wider than those of type $\sqrt{N \log(N)}$. Both the Azuma test and the Darling test have a much wider area $\mathcal{NC}$ than the other tests,

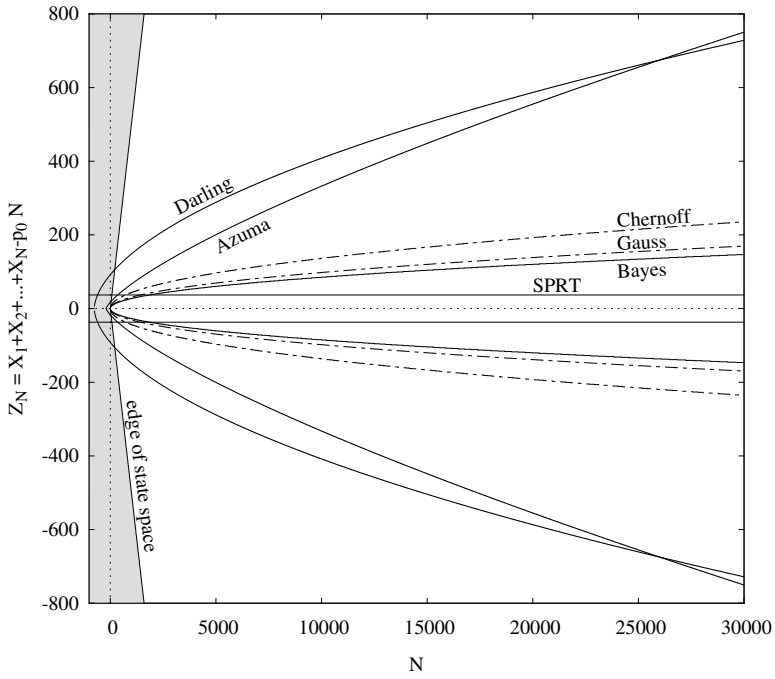


Figure 2.4: Critical regions, $\pi_0 = \frac{1}{2}$, $\delta = \frac{1}{100}$, $\gamma = \frac{1}{10}$. Solid lines are used for the sequential tests, dash-dotted lines for the fixed sample size tests.

which is the price they pay for not risking an inconclusive termination, and for not requiring an indifference region. When we compare the areas $\mathcal{NC}$ of the Gauss test, the Chernoff test and the Bayes test, we see that the Gauss and Bayes tests are strongly similar, which leads to the interesting observation that a sequential version of the Gauss test can satisfy the property of correctness when we assume that π is sampled according to a prior measure $\mathbb{G}$. The Chernoff test behaves similarly to the other two tests, but its area $\mathcal{NC}$ is wider. The area $\mathcal{NC}$ of the SPRT does not widen at all.

Figure 2.5 is similar to Figure 2.4 but with π_0 set to $\frac{1}{20}$. We mention two differences. First, the Gauss test's area $\mathcal{NC}$ is less broad due to the smaller variance under the null hypothesis. Second, the boundaries of the SPRT are no longer constants, but drift downward (see (2.13) and (2.14)).

2.4.2 Simulation Results

In this section, we compare the tests discussed in this chapter by empirically evaluating their performance for a range of underlying parameter values. Since we only compare different statistical tests, we do not need to consider the simulation aspect

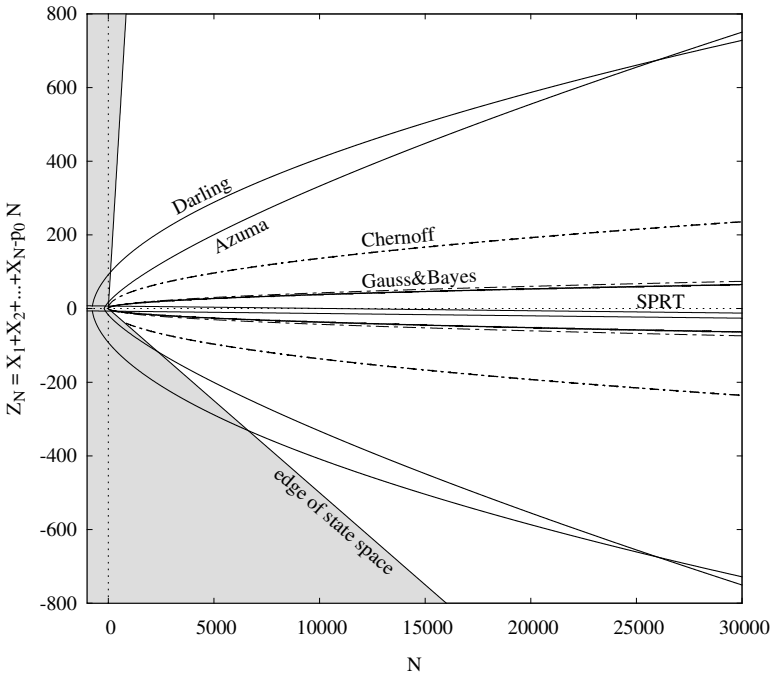


Figure 2.5: Critical regions, $\pi_0 = \frac{1}{20}$, $\delta = \frac{1}{100}$, $\gamma = \frac{1}{10}$. Solid lines are used for the sequential tests, dash-dotted lines for the fixed sample size tests.

of statistical model checking. Accordingly, we let our computer program directly draw samples from a Bernoulli distribution with (known) parameter π. With π chosen, the remaining parameter to be chosen is δ for the SPRT, $\mathbb{G}$ for the Bayes test and γ for the other tests. In all cases $\alpha = \beta_1 = \beta_2 = 0.05$.

For each test we estimate the following metrics:

1. ρ, the probability that a test accepts the right hypothesis, used as a measure for the *correctness* (the higher the better);

2. υ, the probability that a test proves inconclusive, used as a measure for the *power* (the lower the better);

3. η, the expected number of samples drawn before the test is concluded, used as a measure for the *efficiency* (the lower the better).

The procedure is as follows: we conduct each test $1\,000$ times, let $\hat{\rho}$ be the fraction of correct conclusions, $\hat{\upsilon}$ the fraction of tests that remained inconclusive (where for the sequential tests, we set a 60 second time bound) and $\hat{\eta}$ be the average number of samples drawn. In Tables 2.6, 2.7 and 2.8, we display these estimates plus/minus

the half-width of a 95%-CI around the estimate. In Tables 2.6 and 2.7 we have set $\pi_0 = \frac{1}{2}$; the only difference between these two tables is the choice of $|\pi - \pi_0|$, which equals 0.1 for the former and 0.001 for the latter. For Table 2.8 we have set $\pi = \frac{1}{5}$ and $|\pi - \pi_0| = 0.01$. The rows in bold indicate that the guess γ (or indifference level δ) is exactly equal to $|\pi - \pi_0|$.

Test	γ (or δ)	$\hat{\rho}$	$\hat{\upsilon}$	$\hat{\eta}$
Gauss	**0.1**	$\mathbf{0.933 \pm 0.015}$	$\mathbf{0.067 \pm 0.015}$	$\mathbf{3.19 \cdot 10^2}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$3.25 \cdot 10^4$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$3.25 \cdot 10^6$
SPRT	**0.1**	$\mathbf{0.965 \pm 0.011}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(3.62 \pm 0.15) \cdot 10^1}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$(3.65 \pm 0.05) \cdot 10^2$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$(3.69 \pm 0.02) \cdot 10^3$
Chernoff	**0.1**	$\mathbf{0.992 \pm 0.006}$	$\mathbf{0.008 \pm 0.006}$	$\mathbf{6.67 \cdot 10^2}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$6.67 \cdot 10^4$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$6.67 \cdot 10^6$
Bayes	$a = b = 1$	0.957 ± 0.013	0.0 ± 0.0	$(4.62 \pm 0.34) \cdot 10^1$
	$a = b = 10$	0.998 ± 0.003	0.0 ± 0.0	$(7.77 \pm 0.39) \cdot 10^1$
	$a = b = 100$	1.0 ± 0.0	0.0 ± 0.0	$(1.57 \pm 0.05) \cdot 10^2$
Azuma	**0.1**	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(4.98 \pm 0.13) \cdot 10^2}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$(2.28 \pm 0.02) \cdot 10^3$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$(2.12 \pm 0.00) \cdot 10^5$
Darling	**0.1**	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(1.26 \pm 0.02) \cdot 10^3}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$(1.59 \pm 0.02) \cdot 10^3$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$(2.08 \pm 0.03) \cdot 10^3$

Table 2.6: $\pi_0 = 0.5$, $\pi = 0.4$

The number of samples needed for the *Gauss* test grows inversely proportional to the square root of γ. Because the Gauss test is a fixed sample size test, $\hat{\eta}$ has no variance. The main drawback is that if γ is considerably larger than $|\pi - \pi_0|$, the Gauss test will almost never draw a conclusion. This is witnessed by $\hat{\upsilon} \gg 0$, seen particularly in Table 2.7.

The *SPRT* is the most efficient when δ is picked just right; in each table, its value $\hat{\eta}$ is the lowest among all tests that satisfy correctness (not counting the Bayes test which has different assumptions). However, its performance degrades sharply when δ is chosen larger than $|\pi - \pi_0|$. In Table 2.7, the CI for $\hat{\rho}$ contains $\frac{1}{2}$ when δ is large, which is the worst level of ρ that a test can satisfy (after all, if the correctness was even lower one could always use the opposite result of the test and obtain a correctness that is $> \frac{1}{2}$). The average number of samples needed seems to grow inversely proportional to δ.

Both the *Azuma* and *Darling* tests are very conservative: they have a $\hat{\rho}$ of well over 95%. The average total sample sizes for the Azuma and Darling tests compare reasonably well to the approximations of Table 2.3. When the guess is (almost) correct, the Azuma test is more efficient than the Darling test. However, if γ is

taken to be considerably larger than $|\pi - \pi_0|$, the number of samples needed for the Azuma test blows up, while the Darling test remains remarkably insensitive to the model parameters, as can been seen in all tables. The Azuma result $\hat{v} \approx 1$ in Table 2.7 means that the Azuma test did not draw a conclusion within a 60 second time period.

We see in general that the *Chernoff* test requires more samples than the Gauss test; in Table 2.8, for which π_0 equals $\frac{1}{5}$ instead of $\frac{1}{2}$, the difference between the sample sizes of the Gauss and Chernoff tests is larger than in Tables 2.6 and 2.7. This is consistent with the discussion of Section 2.2.3. The bound on the probability of error of the second type for the Chernoff test appears to be rather loose; when the guess γ is correct, the estimate for the probability of inconclusive termination $\hat{v}$ seems to be well below $\beta_2 = 0.05$ in Tables 2.6 and 2.7.

The *Bayes* behaves fundamentally different from the other tests. It requires that its prior $\mathbb{G}$ equals the true distribution from which π is drawn for its bounds on the error probabilities to hold. The true distribution is in this section a degenerate distribution on π_0. The uniform prior (i.e., $a = b = 1$) performs adequately in Table 2.6, but when π is very close to π_0 (e.g., in Table 2.7) or when π_0 does not equal $\frac{1}{2}$ (e.g., in Table 2.8), its performance is unsatisfactory. For $a = b = 100$, we get more conservative results when $\pi_0 = \frac{1}{2}$, but very bad results when $\pi_0 = \frac{1}{5}$. All of this confirms the discussion at the end of Section 2.2.4. In general, it appears to be best to choose a and b such that the mean $a/(a+b)$ of the resulting Beta distribution equals π_0, and that the higher a and b are chosen, the lower the probability of error will be. It is unclear how to determine '*optimal*' choices for a and b, as they lack a clear interpretation in terms of the error probabilities. This ambiguity is an argument against use of the method.

Finally, when we compare the computational complexity of the methods, the fixed sample size tests tend to be much less expensive due to the fact they have to compute the boundary of $\mathcal{NC}$ only once, when the samples have been drawn, whereas the sequential tests require that this happens after each sample. The Chernoff test performs the best overall: the most expensive procedures to be carried out are to evaluate a logarithm three times in order to determine N, and then after N samples have been drawn it needs to compare the sample mean to a constant which requires the evaluation of a logarithm and a square root. The Gauss test performs second best: it requires a numerical root-finding procedure to find N, and exponentiation and evaluation of the inverse of the standard normal distribution function to carry out the test after drawing the samples. Among the sequential tests, the SPRT is the most computationally inexpensive: after each sample N is drawn, the value of the process Z_N only has to be compared to linear boundaries. The Azuma and Darling tests are also computationally easy; the most expensive procedures are the evaluation of two square roots for the former and the evaluation of a square root and a logarithm for the latter. The Bayes test is the most expensive; it requires the inversion of the Beta distribution function, which can be very expensive for some middle-ranged choices of the parameters a and b (for large values one can resort to asymptotic approximations).

Test	γ (or δ)	$\hat{\rho}$	$\hat{v}$	$\hat{\eta}$
Gauss	0.1	0.031 ± 0.011	0.948 ± 0.014	$3.19{\cdot}10^{2}$
	0.01	0.058 ± 0.014	0.933 ± 0.015	$3.25{\cdot}10^{4}$
	0.001	$\mathbf{0.952 \pm 0.013}$	$\mathbf{0.048 \pm 0.013}$	$\mathbf{3.25{\cdot}10^{6}}$
SPRT	0.1	0.505 ± 0.031	0.0 ± 0.0	$(6.31 \pm 0.33){\cdot}10^{1}$
	0.01	0.567 ± 0.031	0.0 ± 0.0	$(5.37 \pm 0.03){\cdot}10^{3}$
	0.001	$\mathbf{0.957 \pm 0.013}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(3.35 \pm 0.02){\cdot}10^{5}}$
Chernoff	0.1	0.004 ± 0.004	0.995 ± 0.004	$6.67{\cdot}10^{2}$
	0.01	0.019 ± 0.008	0.981 ± 0.008	$6.67{\cdot}10^{4}$
	0.001	$\mathbf{0.987 \pm 0.007}$	$\mathbf{0.013 \pm 0.007}$	$\mathbf{6.67{\cdot}10^{6}}$
Bayes	$a = b = 1$	0.575 ± 0.031	0.0 ± 0.0	$(5.40 \pm 1.80){\cdot}10^{4}$
	$a = b = 10$	0.636 ± 0.030	0.0 ± 0.0	$(1.05 \pm 0.22){\cdot}10^{5}$
	$a = b = 100$	0.729 ± 0.028	0.0 ± 0.0	$(1.55 \pm 0.24){\cdot}10^{5}$
	$a = b = 1000$	0.816 ± 0.024	0.0 ± 0.0	$(2.88 \pm 0.31){\cdot}10^{5}$
Azuma	0.1	0.0 ± 0.0	1.0 ± 0.0	$(6.67 \pm 0.01){\cdot}10^{8}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$(1.40 \pm 0.02){\cdot}10^{8}$
	0.001	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(4.63 \pm 0.13){\cdot}10^{6}}$
Darling	0.1	1.0 ± 0.0	0.0 ± 0.0	$(2.81 \pm 0.03){\cdot}10^{7}$
	0.01	1.0 ± 0.0	0.0 ± 0.0	$(2.42 \pm 0.03){\cdot}10^{7}$
	0.001	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(2.25 \pm 0.03){\cdot}10^{7}}$

Table 2.7: $\pi_0 = 0.5$, $\pi = 0.499$

2.5 Steady-State Measures

So far, we have restricted ourselves to verifying properties of the form $\mathcal{P}_{>p}(\phi)$, where ϕ is some path formula and $\mathcal{P}$ is the probabilistic operator found in logics such as PCTL and CSL (see Section 1.1.2). These logics also contain the *steady-state* operator $\mathcal{S}_{>\pi_0}(\Phi)$, where Φ is a CTL-state formula. In this section we focus on the steady-state operator. We say that $\mathcal{S}_{>\pi_0}(\Phi)$ is satisfied when

$$v_\Phi \triangleq \sum_{x \models \Phi} v(x) > \pi_0, \tag{2.26}$$

where $v(x)$ is the steady-state probability of being in state x of the Markov chain.

As is the case for the $\mathcal{P}$-operator, one might find it necessary to resort to statistical techniques if the state space is too large to compute the steady-state probabilities numerically. Statistical techniques for estimating or testing hypotheses regarding steady-state probabilities are more complicated than their analogues for transient measures. The most obvious estimate for v_Φ is the long-run average fraction of time spent in states that satisfy Φ. The problem is how to decide that the run is long enough to be used for a long-run average.

There are several ways to solve this problem, of which we first discuss the *batch means method*, followed by the *regeneration method* and *perfect simulation*. The main

Test	γ (or δ)	$\hat{\rho}$	$\hat{v}$	$\hat{\eta}$
Gauss	0.1	0.036 ± 0.012	0.953 ± 0.013	$1.64{\cdot}10^2$
	0.01	$\mathbf{0.946 \pm 0.014}$	$\mathbf{0.054 \pm 0.014}$	$\mathbf{2.04{\cdot}10^4}$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$2.39{\cdot}10^6$
SPRT	0.1	0.489 ± 0.031	0.0 ± 0.0	$(3.70 \pm 0.17){\cdot}10^1$
	0.01	$\mathbf{0.949 \pm 0.014}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(2.19 \pm 0.10){\cdot}10^3}$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$(2.39 \pm 0.03){\cdot}10^4$
Chernoff	0.1	0.007 ± 0.005	0.993 ± 0.005	$6.67{\cdot}10^2$
	0.01	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{6.67{\cdot}10^4}$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$6.67{\cdot}10^6$
Bayes	$a = b = 1$	0.599 ± 0.030	0.0 ± 0.0	$(5.64 \pm 0.56){\cdot}10^2$
	$a = 1, b = 4$	0.853 ± 0.022	0.0 ± 0.0	$(8.02 \pm 0.63){\cdot}10^2$
	$a = b = 100$	0.0 ± 0.0	0.0 ± 0.0	1.00 ± 0.0
	$a = 100, b = 400$	0.994 ± 0.005	0.0 ± 0.0	$(1.64 \pm 0.04){\cdot}10^3$
Azuma	0.1	1.0 ± 0.0	0.0 ± 0.0	$(1.41 \pm 0.01){\cdot}10^6$
	0.01	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(4.79 \pm 0.10){\cdot}10^4}$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$(2.24 \pm 0.01){\cdot}10^5$
Darling	0.1	1.0 ± 0.0	0.0 ± 0.0	$(2.04 \pm 0.02){\cdot}10^5$
	0.01	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{(1.78 \pm 0.02){\cdot}10^5}$
	0.001	1.0 ± 0.0	0.0 ± 0.0	$(2.10 \pm 0.02){\cdot}10^5$

Table 2.8: $\pi_0 = 0.2$, $\pi = 0.19$

idea behind the batch means method is to divide the total sample into N *batches* of length k. We run one lengthy simulation consisting of $N \cdot k$ steps (with N yet to be determined) and a sequence $s_1, \ldots, s_{Nk}$ of visited states. Let $\mathbf{1}_\Phi(s_i)$ equal 1 if the i-th state observed in the simulation satisfied Φ and 0 otherwise. Then a consistent estimator of v_Φ would be

$$S'_N \triangleq \frac{1}{Nk} \sum_{i=1}^{Nk} \mathbf{1}_\Phi(s_i),$$

(where the apostrophe serves to distinguish from the S_N of Section 2.1.2) were it not for the fact that the random variables underlying the realisations $\mathbf{1}_\Phi(s_i)$ are not mutually independent. For the j-th batch, one can compute the *batch mean*

$$\bar{y}_j(\Phi) = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}_\Phi(s_{i+(j-1)k}).$$

The N batch means then become mutually independent and normally distributed as $k \to \infty$. With $Z'_N \triangleq S'_N - N\pi_0$, this means that for large values of k we can use the Central Limit Theorem to construct a $(1-\alpha)$-CI for Z'_N.

Since we can construct confidence intervals, we can also construct a fixed sample size hypothesis test as follows: continue sampling until we have N batches and construct the $(1-\alpha)$-CI for Z'_N. We then reject the null hypothesis if 0 is outside the

CI. If so, accept H_{+1} if $Z'_N > 0$ and H_{-1} otherwise. This procedure forms the basis of the fixed sample size test for statistical model checking proposed in Section 6.3 of [117] (see also the discussion of the Gauss test at the end of Section 2.2.1), and this test is then generalised to settings in which the Markov chain consists of more than one strongly connected component.

When we try to use a sequential test, the question is to what extent we can apply the sequential tests discussed previously in this chapter. Clearly, Z'_N is longer the sum of random variables defined by (2.1), so the SPRT and Bayes test will not work in this situation. However, we have approximate normality if k is large enough, which means that we can apply the Darling test under the assumption that the steps are normal. If, following the application of the Darling test, a normality test such as the Shapiro-Wilk test [52] fails to reject normality, we can a posteriori conclude that the test went well. Also, $Z'_N - Z'_{n-1}$ is bounded so the Azuma test can also be applied. The only issue then is the independence of the steps - this can also be tested a posteriori using a test such as the Spearman correlation test [52]. The remaining problem is then the selection of the batch size b [70], for which we have no formal algorithm.

For the other main method, called the *regenerative* approach, it is also possible to construct a process Z''_N that fits in the framework of Section 2.1.[16] However, in this case we do not have *any* information about the probability distribution of the step sizes $Z''_N - Z''_{N-1}$ so the Darling test, for which we need this knowledge, will not work. As they are not even bounded, the Azuma test will also not work.

As for other existing sequential tests for steady-state properties, we mention the tests proposed in [37] and evaluated in [36]. This test avoids the problem of warming up the simulation (called the burn-in time in these papers) by using *perfect simulation* [86] to sample directly from the steady-state distribution v. Perfect simulation (also called *coupling from the past*) is a technique that can be applied when the Markov chain satisfies one of several technical conditions — the most common being *monotonicity* — that imply that samples can be drawn from the steady-state distribution within finite time. In this way, tests that requires that the samples are drawn from a random variable with a Bernoulli distribution (e.g., the SPRT) can be applied. However, in [36] monotonicity is required for the test to be applicable, and this property does not hold for many models and if it does, it is non-trivial to prove. Furthermore, the dependence of the runtime of the method on the size of the state space (a "*functional*" dependence cf. [36]) is unclear.

2.6 Conclusions and Discussion

We have presented a common framework that allows the methods proposed earlier in the statistical model checking literature to be compared in a mathematically

[16] The idea behind this method is to repeatedly simulate *regeneration cycles* that start and end when a predetermined regeneration state is entered. The process Z''_N can then be defined as the ratio between the time spent in Φ during the first N regeneration cycles and the total time spent during the first N regeneration cycles. For more information, see [117].

solid, yet intuitive manner. Previously, when these methods (e.g., the SPRT and the Chernoff test) were built into model checking tools such as PRISM and UPPAAL, they were often implemented completely parallel to one another with little information given about the subtle differences between the methods and their parameters. We believe that our contribution aids the general understanding of these methods, reducing the likelihood of interpretative errors.

We have also introduced two tests for statistical model checking, the Azuma test based on new analysis and the Darling test based on a power one test introduced in [28]. We compared the performance of these tests to the four most prevalent existing tests in the literature. The appeal of the two new tests is their safety: neither their correctness nor their power depends on parameters that require (correct) prior information about the system model. Also, the new tests do not have the SPRT's fundamental shortcoming that it will not be clear from the result of the test whether the assumptions of the test have been valid. Hence, a good guess for the true probability is only needed for efficiency, in particular for the Azuma test. The efficiency of the Darling method is fairly insensitive to the guess.

The two new tests constitute a powerful addition to the toolset of the investigator who is uncertain about which parameters to give as input to her statistical model checker. If the investigator has a good reason to choose a certain indifference level *and* believes the actual value $|\pi - \pi_0|$ to be close (say, up to two orders of magnitude) to this indifference level, then the SPRT remains her method of choice. In all other cases, she is faced with the possible loss of efficiency or, even worse, correctness. The two new tests provide safe and solid alternatives. A single fixed sample size test of course remains possible if one is willing to accept the risk of a inconclusive outcome.

Furthermore, the Azuma test is motivated by Theorem 2.1, which can be applied to any random walk. Clearly, when model checking $\mathcal{P}_{>\pi_0}(\phi)$ the samples drawn form a random walk, but there are other settings in which Theorem 2.1 can be applied. Using Theorem 2.1 we proposed a sequential test for model checking the steady-state operator $\mathcal{S}_{>\pi_0}(\Phi)$, although this test still requires the batch size to be chosen as a parameter.

2.6.1 Hypothesis Testing for Importance Sampling

While this chapter has been about hypothesis testing, the next four chapters are about importance sampling. It is an obvious question to ask how these topics connect. As mentioned in the introduction, in the following chapters we primarily concern ourselves with making quantitative statements, meaning that we demonstrate how to efficiently estimate the probabilities of interest using importance sampling (see Section 1.3). Given such an estimate, we use the CLT to construct a confidence interval (CI) as given either in (1.4) or (1.6). Note that in this setting we *have* to use the CLT to construct a CI because we typically have no idea about the probability distribution of the likelihood ratios of the sample paths as defined in (1.8). Using the CI we can carry out the Gauss test, although the CI is only an approximation,

the quality typically depending on the change of measure. The Chernoff-Hoefding bound can *only* be applied to construct a true confidence interval if the likelihood ratios are upper bounded — only in this case can we also apply the Chernoff test. The fact that we have no information about these distributions also rules out the Darling test, the Bayes test and the SPRT. Hence, the only sequential test of the ones discussed in this chapter that can be applied to importance sampling estimators is the Azuma test, again, *if* the likelihood ratios can be upper bounded.

In this context we mention [11], in which a change of measure is proposed in which the likelihood ratios are indeed bounded. However, the application of the technique of this paper requires that a model is found that is *coupled* to the original model and which is then proved to satisfy a technical property (i.e., being a so-called *reduction of guaranteed variance*). However, as with the monotonicity that is required to apply the test of [37] (which is also based on coupling), proving this condition is non-trivial.

For the changes of measure that we consider in the rest of this thesis, it will *not* hold that the likelihood ratios are bounded. However, if we can upper bound the effect of a single transition on the likelihood ratio (which is a reasonable assumption in many settings where the behaviour of the model is given through a high-level description language), then we could apply the following procedure: sample using importance sampling *until* the likelihood ratio has become so large that the following transition could raise it above some given upper limit. In this case we stop simulating under importance sampling and continue using standard Monte Carlo. This need not have a big effect on the efficiency of the IS-estimator; as we discuss in the remainder of this thesis, our IS-methods work by increasing the likelihood of sampling paths that are *dominant* in some sense, and when transitions are drawn that correspond to behaviour along the dominant paths the likelihood ratio will *decrease*. This idea has however not been fully explored and hence remains a subject for further research.

CHAPTER 3

Rarity Regimes in the Birth-Death Process

Chapter 3 is the first of four chapters on rare event simulation using importance sampling, with the specific focus of this chapter being on the model class of (parallel) birth-death processes. The limited scope of this class allows for an easy and intuitive explanation of the core concepts of *rarity regimes* and *dominant paths*. A rarity regime is a combination of parameters and corresponding limiting values such that if the parameter values approach their limits, the probability of interest goes to zero and — importantly — that in the increasingly unlikely case that an event of interest occurs, the way in which it happens will increasingly resemble asymptotic behaviour. The dominant paths in a rarity regime are the paths that come to dominate the event of interest probabilistically, i.e., the total probability of the dominant paths approaches the probability of interest in the limit associated with the rarity regime.

The dominant paths are of great interest to us if we want to use importance sampling. After all, the importance sampling approach described in Section 1.3.5 requires that we substitute an approximation for the probability of interest in each state into (1.12) or (1.13), yielding an importance sampling change of measure. If the dominant paths come to dominate the probability of interest, a good approximation for the probability of interest from a state x is the probability of the dominant paths from x. Birth-death processes are a good model class with which to illustrate this approach because the dominant paths in this model class have a simple form for several important regimes, as we will see later in this chapter. Despite the limited scope of the birth-death process as a modelling class, it includes well-known benchmark cases such as the M/M/c queueing system [46] and reliability models consisting of independent component types.

In this chapter we present a variety of results. We present an overview of performance measures and rarity regimes commonly used in the rare event simulation literature, and discuss their meaning in the context of the birth-death process. Furthermore, we zoom in on a specific regime that is relevant in the setting of Markov reward models and discuss how efficient simulation of sojourn times in the states of the Markov chain in this setting is different from settings for which forcing (as discussed in Section 1.3.4) suffices — we then present a simulation algorithm that works well in this setting. Also, this section serves as a prelude to Chapter 4, in which we consider networks of birth-death processes and which uses several of the performance measures and rarity regimes discussed in this chapter.

The outline of this chapter is as follows. In Section 3.1 we introduce the birth-death process, discuss the main performance metrics and explain the concept of a dominant path. In Section 3.2, we highlight the parameter settings in which the performance metrics start to involve rare events, and discuss the impact of the different regimes on the simulation procedure. In Section 3.3 we focus on drawing sojourn times in a pure birth process in the rarity regime of Section 3.2 (i.e., *'large mission times'*). Section 3.4 concludes the chapter.

3.1 The Birth-Death Process

The birth-death process is a stochastic process defined on a one-dimensional state space such that in all states there are at most two transitions possible: one to the right (the *'birth'*) and one to the left (the *'death'*). As the name suggests, the birth-death process is a widely used formalism for population models used in fields such as biology [82] and demography, but it has applications in many more other areas, e.g., queueing theory and performance analysis. In Section 3.1.1, we give a formal definition of a birth-death process as a CTMC and explain several real-world interpretations. In Section 3.1.2, we discuss several performance measures that are of interest in a birth-death process. In Section 3.1.3 we discuss the dominant paths in a birth-death process.

3.1.1 Definition

The birth-death process is a CTMC with state space $\mathcal{X} = (x_i)_{i\in\mathbb{N}}$ and at most two transitions enabled in each state: the λ-transition which takes a state x_i to x_{i+1} $\forall i \geq 0$, and the μ-transition which takes a state x_i to x_{i-1} $\forall i \geq 1$. The rates can be state-dependent, meaning that the λ-transition has rate $\lambda(x) \geq 0$ in state x and the μ-transition has rate $\mu(x) \geq 0$ in state x. A depiction of a birth-death process is given in Figure 3.1.

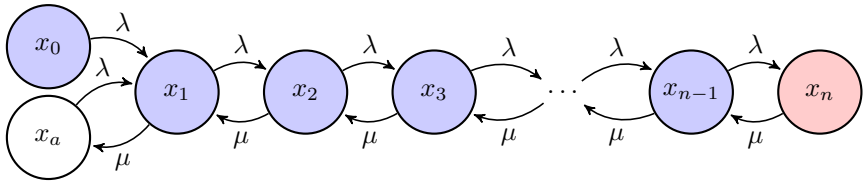


Figure 3.1: An M/M/1/n queue [46], depicted as a CTMC. The initial state is x_0; since we are often interested in events involving a return to the initial state, we duplicate x_0 and make it an a-state (the result is x_a). The buffer overflow state x_n is a goal state.

The interpretation of the state and the transitions depends on the application setting. In a biological setting, the birth-death process is a population model in which the state is the number of living organisms, and where the λ-transition represents the *birth* and the μ-transition represents the *death* of an organism. In queueing, the

state of the process is the number of customers in the queue or the system and the λ-transition represents an *arrival* and the μ-transition the completion of service or *departure* of a customer. In applications in reliability engineering, the state is the number of active or spare components of a type — the λ-transition then represents the *failure* and the μ-transition represents the *repair* of a single component.

Typically, the rates of the λ-transition are (much) lower than those of the μ-transition, especially for states to the '*right*', i.e., x_i with large values of i.[17] As a result, the rightmost states are not visited often and we are interested in (rare) events involving these states. To formalise these events, let there be some constant $n \in \mathbb{N}$ such that x_n and all states to the right of it be b-states. We let x_0 be the initial state; if we are interested in events involving a return to the initial state, we add an a-state x_a to which all the transitions to x_0 are routed. The performance measures usually involve some probability that becomes smaller as it becomes less likely to reach the goal set when one or more parameters attain increasingly extreme values.

Note that we can view the birth-death process as an instance of a stochastic Petri nets (SPN; see Section 1.1.3), with one place and two transitions. In a reliability setting, the number of tokens then represents the number of broken components and the two transitions represent component failures and repairs respectively.

3.1.2 Performance Measures

We focus on three performance measures that are commonly used in reliability engineering, and discuss all of them in the context of a single birth-death process. The first is the *(transient) unreliability*, the probability of hitting x_n before some fixed time bound $\bar{\tau}$. The second is the *conditional unreliability*, the probability of hitting x_n before the taboo state x_a. The third is the *unavailability*, the steady-state probability of being in a goal state.

The (Transient) Unreliability

The first performance measure that we discuss is the unreliability, although it is the most challenging one from a theoretical point of view. The reliability is a time-bounded measure such that a system is highly reliable according to this measure if it is unlikely that the system breaks down within some given execution period. We assume that the system is highly reliable, making the unreliability a rare event probability. We distinguish three main variants.

- The first is the most common one, namely the probability of observing a goal state before time $\bar{\tau}$ (the *mission time*). This is written in CSL (see Section 1.1.2) as $\Diamond^{[0,\bar{\tau}]} b$.
- Second we have the probability of observing $\psi^{\bar{\tau}}$, the event of hitting a goal state before the taboo state *and* before time $\bar{\tau}$, written in CSL as $\neg a \,\mathsf{U}^{[0,\bar{\tau}]} b$. Because we have explicitly defined x_a in birth-death processes to be different from the initial state this is not the same as $\Diamond^{[0,\bar{\tau}]} b$.

[17] This is also necessary for the steady-state distribution to exist.

- Finally, one could be interested in the probability of the rare event that the system breaks down only *after* the mission time has passed. It may be counter-intuitive to say that this probability is low in a highly reliable system, but this setting has applications in Markov reward models as we explain in Section 3.2.4 (this section is also the only section in which this property is further considered). In CSL, this is written as $\neg b \,\mathsf{U}^{(\bar{\tau},\infty)}\, b$.

In the remainder of this subsection we focus on $\mathbb{P}(\psi^{\bar{\tau}})$ for the sake of brevity. Note that in the setting of $\psi^{\bar{\tau}}$ we can make the states x_a and x_b absorbing, which they are not in Figure 3.1.

All probabilities discussed previously can be expressed by the distribution function of a phase-type distributed random variable. The easiest way to write distribution functions of this form is by using a matrix exponential [39]. We give its form explicitly as we need it for future reference: with $\eta(x) \triangleq \lambda(x) + \mu(x)$, we write the generator matrix of the birth-death process as

$$\begin{pmatrix} S & \boldsymbol{\lambda} \\ \mathbf{0} & 0 \end{pmatrix}, \tag{3.1}$$

with $\mathbf{0} = \begin{pmatrix} 0 & \cdots & 0 \end{pmatrix}$, $\boldsymbol{\lambda} = \begin{pmatrix} 0 & \cdots & 0 & \lambda(x_{n-1}) \end{pmatrix}^{\mathrm{T}}$ and where

$$S \triangleq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & -\lambda(x_0) & \lambda(x_0) & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ \mu(x_1) & 0 & -\eta(x_1) & \lambda(x_1) & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \mu(x_2) & -\eta(x_2) & \lambda(x_2) & \ddots & & & \vdots \\ 0 & 0 & 0 & \mu(x_3) & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & \lambda(x_{n-3}) & 0 \\ \vdots & \vdots & \vdots & & & \ddots & \mu(x_{n-2}) & -\eta(x_{n-2}) & \lambda(x_{n-2}) \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 & \mu(x_{n-1}) & -\eta(x_{n-1}) \end{pmatrix}.$$

In S, the first row (and column) corresponds to the (now absorbing) state x_a and the next n rows to x_0 to x_{n-1}. There is no row or column that corresponds to x_b in S, but the final row and column in the generator matrix of (3.1) do. With α the initial distribution given as a (row) vector, i.e., a unit vector with a 1 in the second entry, and ι a column vector of only ones, the unreliability is then given by

$$\mathbb{P}_x(\psi^{\bar{\tau}}) = 1 - \alpha e^{S\bar{\tau}} \iota. \tag{3.2}$$

The idea behind this expression is as follows: with $\eta^* = \max_{x \in \mathcal{X}} \eta(x)$, writing down the Maclaurin series expansion of the matrix exponential yields an infinite sum in which the ith element corresponds to the Poisson probability that i jumps with rate η^* occur within $\bar{\tau}$ time units, multiplied by the transition probability matrix of the corresponding (uniformised) DTMC to the ith power. Multiplying this from the

left by α gives the probability distribution over all states in $\mathcal{X}$ of being there at time $\bar{\tau}$, starting in x_0. Multiplying by ι from the right then gives the sum of these probabilities for all states that do not equal x_b, and 1 minus this probability is the probability of having reached the absorbing state x_b within the mission time.

Alternatively, the probability of interest can be written by conditioning on the sample path and $\neg a \,\mathsf{U}\, b$, i.e.,

$$\mathbb{P}_x(\psi^{\bar{\tau}}) = \sum_{\substack{\omega: x_0^\omega = x, \\ \omega \vDash \neg a \,\mathsf{U}\, b}} \mathbb{P}(\omega) \cdot \mathbb{P}(\psi^{\bar{\tau}} | \omega). \tag{3.3}$$

In this expression, $\mathbb{P}(\omega)$ is a simple product of transition probabilities and $\mathbb{P}(\psi^{\bar{\tau}}|\omega)$ is the probability that the sum of a number of exponentially distributed random variables is smaller (or bigger) than $\bar{\tau}$. There exist analytical expressions for $\mathbb{P}(\psi^{\bar{\tau}}|\omega)$ [3], but their numerical evaluation is very cumbersome if the rates are not all equal or all different. However, one could use approximative techniques such as Gauss-Seidel (also mentioned in Section 1.1.4) to obtain these probabilities. If the sample paths typically observed in a simulation run are not too large, a simulation technique could be to first generate the sample paths and to compute $\mathbb{P}(\psi^{\bar{\tau}}|\omega)$ numerically using iterative methods or to independently estimate this probability using importance sampling. This will be discussed further in Sections 3.2.3 and 3.2.4.

The Conditional Unreliability

In the setting of a single birth-death process, the conditional unreliability is simply the probability of hitting a high level before the lowest level — i.e., $\mathbb{P}(\psi)$. It is equal to $\mathbb{P}(\psi^{\bar{\tau}})$ as discussed in the previous section with $\bar{\tau} = \infty$. In a queue, this is the probability of reaching an overflow state before the queue empties, starting from the empty state. In queueing, the time interval between the arrival of a customer to an empty queue and the queue again becoming empty is called a *busy period*,[18] and the conditional unreliability is, hence, the probability of reaching a high level during a single busy period. In a reliability setting, the conditional unreliability is the probability of seeing a large number of components break down (typically a number large enough to cause the whole system to fail) before all components are up again.

The conditional unreliability in x_i, $i = 2, \ldots, n-1$, is given by the recursion relation

$$\mathbb{P}_{x_i}(\psi) = \frac{\lambda(x_i)}{\lambda(x_i) + \mu(x_i)} \mathbb{P}_{x_{i+1}}(\psi) + \frac{\mu(x_i)}{\lambda(x_i) + \mu(x_i)} \mathbb{P}_{x_{i-1}}(\psi), \tag{3.4}$$

which together with the conditions $\mathbb{P}_{x_a}(\psi) = 0$, $\mathbb{P}_{x_n}(\psi) = 1$, $\mathbb{P}_{x_0}(\psi) = \mathbb{P}_{x_1}(\psi)$ and $\mathbb{P}_{x_1}(\psi) = \frac{\lambda(x_1)}{\lambda(x_1)+\mu(x_1)} \mathbb{P}_{x_2}(\psi)$ allows the conditional unreliability to be calculated in each state. When the rates are constant, i.e., $\lambda(x) \equiv \lambda$ and $\mu(x) \equiv \mu \; \forall x \in \mathcal{X}$,

[18] Note that in Section 1.2.1 we introduced the notion of a busy *cycle*, which is the busy period plus the time spent in the regeneration state. Throughout the remainder of this chapter we use the concept of the busy cycle as defined in Section 1.2.1.

the calculation of the conditional unreliability is equivalent to the *Gambler's Ruin Problem* (also known as *Gambler's Fortune* [99]). This problem has a simple closed form solution given by

$$\mathbb{P}_{x_i}(\psi) = \frac{(\frac{\mu}{\lambda})^i - 1}{(\frac{\mu}{\lambda})^n - 1}, \ \forall i = 1, \ldots, n. \tag{3.5}$$

Substituting this expression into (1.13), one obtains the following expression for the zero variance transition probabilities:

$$q_{x_i x_{i+1}} = \frac{\mu^{i+1} - \lambda^{i+1}}{(\mu + \lambda)(\mu^i - \lambda^i)} \ \text{ and } \ q_{x_i x_{i-1}} = 1 - q_{x_i x_{i+1}} \ \forall i = 1, \ldots, n.$$

As noted in [30], the zero variance measure does not depend on n in this case.

When there exist different states x and z such that $\lambda(x) \neq \lambda(z)$ or $\mu(x) \neq \mu(z)$, we cannot in general know $\mathbb{P}(\psi)$ for a single state x without computing it for all states x_i, $i = 1, \ldots, n-1$. However, we *can* obtain the zero variance transition probabilities starting from x_1 without further knowledge of all other transition probabilities. The reason is that for a measure to produce an estimator with zero variance, it must hold that $L_{\mathbb{Q}}(\omega)$ is the same for each path ω. As such, for $\omega = (x_0^\omega, \ldots, x_i^\omega, \ldots, x_{|\omega|}^\omega)$ and $\omega' = (x_0^\omega, \ldots, x_i^\omega, x_{i-1}^\omega, x_i^\omega, \ldots, x_{|\omega|}^\omega)$ it must hold that $L_{\mathbb{Q}}(\omega) = L_{\mathbb{Q}}(\omega')$, which in turn implies that the likelihood ratio caused by the *cycle* $x_i^\omega \to x_{i-1}^\omega \to x_i^\omega$, given by

$$\frac{p_{x_i^\omega x_{i-1}^\omega} p_{x_{i-1}^\omega x_i^\omega}}{q_{x_i^\omega x_{i-1}^\omega} q_{x_{i-1}^\omega x_i^\omega}},$$

must equal one. Since it must hold that $q_{x_1 x_a} = 0$ as there would otherwise be positive probability of observing a path ω that would not result in the rare event, $q_{x_1 x_2}$ can be computed directly. We can then sequentially compute all remaining zero variance probabilities. Once we are finished, the true probability $\mathbb{P}_{x_j}(\psi)$ can then be computed by

$$\mathbb{P}_{x_j}(\psi) = \prod_{i=j}^{n-1} \frac{p_{x_i x_{i+1}}}{q_{x_i x_{i+1}}},$$

The Unavailability

The availability is a performance measure given by the long-term average fraction of time in which the system is operational, i.e., in a state x_i such that $i < n$. In a highly reliable system, its converse — the *un*availability — is a small probability, namely the steady-state probability of being in a state where the system has failed. The steady-state probability distribution of the system can be expressed using a system of equations — solving this system for the birth-death process yields that the steady-state probability of being in state x_i, $i = 1, 2, \ldots$, is given by

$$v(x_i) = \frac{\lambda(x_a)\lambda(x_1) \cdots \lambda(x_{i-1})}{\mu(x_1)\mu(x_2) \cdots \mu(x_i)} v(x_a),$$

where $v(x)$ is the steady-state probability of being in state x, and

$$v(x_a) = \frac{1}{1 + \sum_{i=1}^{\infty} \frac{\lambda(x_a)\lambda(x_1)\cdots\lambda(x_{i-1})}{\mu(x_1)\mu(x_2)\cdots\mu(x_i)}} \tag{3.6}$$

(see [74]). However, we do not use this information directly as we use the ratio estimator (1.5), and (3.6) does not trivially lead to zero variance transition probabilities for estimating the quantities in this expression. Instead, our approach will more closely resemble importance sampling for the conditional unreliability. The reason is that of the two quantities that appear in (1.5), only the fraction of time spent in the goal set during a busy cycle is the one that is hard to estimate. The problem that we face is that for a typical timed sample path $\boldsymbol{\omega}$ the system will not reach the goal set, meaning that $Z(\boldsymbol{\omega})$ will remain zero. So to estimate this quantity efficiently, we need to increase the probability of hitting the goal set during a busy cycle, which is exactly the same problem as for the conditional unreliability.

The procedure will be as follows: we start in the initial state x_0 and simulate using the same importance sampling procedure that we would use for the conditional unreliability. We stop when we reach system failure and from then on simulate using the old distribution until we reach the regeneration state, namely x_a [50]. Meanwhile, we record the amount of time during which the system was in a failed state.

We note here that it is typically not a good idea to use the same change of measure for both $\mathbb{E}(Z)$ and $\mathbb{E}(D)$. For example, if we use zero variance approximation based on (1.13), then the paths that immediately fall back to the regeneration state x_a before reaching a system failure state are not sampled because $w(x_a) \equiv 0$ for any reasonable approximation w. This has no effect on the unbiasedness of the estimator $\hat{z}$ because paths that immediately fall back to the regeneration state contribute nothing to $\mathbb{E}(Z)$. However, they do contribute heavily to $\mathbb{E}(D)$. Therefore, to avoid bias and inconsistency in $\hat{d}$ we generate two series of runs, one for Z with importance sampling and one for D without importance sampling [45], and substitute them into (1.5).

3.1.3 Straight Paths and Cycles

As defined in the introduction of this chapter, the dominant paths are those paths for which it holds that the probability of interest approaches the sum of the probabilities of the dominant paths in the limit associated with a given rarity regime. Of course, the structure of this set depends on the chosen rarity regime, to be discussed in Section 3.2. However, in most of the rarity regimes the dominant paths have a similar form.

The first obvious candidate for a dominant path in the birth-death process is the straight path, which from state x_i is given by $(x_i, x_{i+1}, \ldots, x_{n-1}, x_n)$. In this setting, the straight path is the most likely path and certainly a dominant path. However, as we will see in Chapter 4, in a *network* of d parallel birth-death processes, there

are d straight paths from any state outside the goal and taboo sets to the goal set (assuming the non-trivial case where in the ith isolated birth-death process the rates are such that with positive probability the ith element of $\vec{x}$ eventually reaches n_i). In this setting the straight path need *not* be the most likely path— e.g., a path that first falls back to x_a by observing deaths in the ith process and then reaches overflow through births in the jth process, $i \neq j$, may be more likely than the straight path in which there are only births in the ith process. This will all be explained in more detail in Chapter 4

In a single birth death process, all paths that are not the straight path contain *cycles*. A cycle is in itself a path, but one which begins and ends in the same state, i.e., a path $(x_0^\omega, x_1^\omega, \ldots, x_{|\omega|}^\omega)$ such that $x_0^\omega = x_{|\omega|}^\omega$. As we will see in Section 3.2.2 (and in Chapter 4), some paths that contain cycles will also be dominant paths, even if they are never the most likely.

3.2 Rarity Regimes in the Birth-Death Process

Having discussed the straight paths, the main question is whether they are dominant, and if so, in which asymptotic setting. As such, the focus of this section is on the different rarity regimes in a birth-death process and the effect of these regimes on the typical way in which rare events occur. Formally, we define a rarity regime to be any asymptotic regime in which the probability of interest goes to zero.

We note here that one can think of an infinite number of rarity regimes, particular one where several parameters reach extreme values with different speeds (e.g., a regime where $\lambda \sim n^2$ and $n \to \infty$). However, we restrict ourselves to regimes that are common in the literature and motivate each regime with its interpretation in a reliability setting. Furthermore, it may be possible to simulate a regime involving one or more parameters by manipulating other parameters, e.g., for the conditional unreliability $\lambda \downarrow 0$ and $\mu \to \infty$ are equivalent.

In this section, we largely intend to provide an intuitive description of the different rarity regimes. As such, some statements that could be made mathematically precise are not. We only discuss the setting where the repair rates and the failure rates are equal in all states (except x_0, in which repairs are not allowed).

3.2.1 The Regimes $\lambda \downarrow 0$ and $\bar{\tau} \downarrow 0$ (Slow Component Failures)

We begin with the setting $\lambda \downarrow 0$ (slow component failures) because the behaviour observed in this setting is similar to the behaviour observed in the models of Chapters 5, 6 and 7. Furthermore, we argue that in the time-bounded setting (involving the unreliability), the behaviour observed in the regime of $\bar{\tau} \downarrow 0$ is similar in the sense that the straight path is the sole dominant path in both settings. We first discuss the conditional unreliability and then the unreliability.

For the conditional unreliability (i.e., $\mathbb{P}(\psi)$), it holds in this regime that the straight path is not just dominant, but that it by itself comprises the entire set of dominant paths. To see this, note that the probability of the straight path state ω_i from x_i is

given by $(\lambda/(\lambda+\mu))^{n-i}$, and that the true probability $\mathbb{P}(\psi)$ is given by (3.5). The ratio $\mathbb{P}(\omega_i)/\mathbb{P}(\psi)$ equals

$$\frac{\left(\frac{\mu}{\lambda+\mu}\right)^n - \left(\frac{\lambda}{\lambda+\mu}\right)^n}{\left(\frac{\mu}{\lambda+\mu}\right)^i - \left(\frac{\lambda}{\lambda+\mu}\right)^i}$$

which goes to 1 as λ goes to zero. The intuition behind this is that all other paths contain cycles in the '*interior*' (i.e., somewhere between x_1 and x_{n-1}), and as these cycles contain the transition with rate λ their probability vanishes as $\lambda \downarrow 0$. Hence, for small values of λ the probability of the straight path is a very good approximation for the true probability, and can be expected to produce an estimator with decent performance. We will see in Chapter 5, which also tackles this problem in a much more general setting, that this approximation in fact produces an estimator with the very desirable property of vanishing relative error.

For the unreliability we distinguish between $\neg a \,\mathsf{U}^{[0,\bar{\tau}]} b$ and $\Diamond^{[0,\bar{\tau}]} b$. In the former setting, we can use the decomposition (3.3); in this expression it is clearly visible that if the time bound is so large that $\mathbb{P}(\psi^{\bar{\tau}}|\omega) = 1$ we are back in the setting of the conditional unreliability. Note that, since all exit rates are equal in the relevant part of the state space, $\mathbb{P}(\psi^{\bar{\tau}}|\omega)$ has an Erlang distribution with rate $\lambda + \mu \approx \mu$ and its shape parameter equal to the number of steps needed for ω. For the straight path ω_0, which was already the only dominant path in the setting of the conditional unreliability, $\mathbb{P}(\psi^{\bar{\tau}}|\omega_0)$ is higher than the related probability of all other paths, meaning that in this setting it remains the only dominant path.

In the setting $\Diamond^{[0,\bar{\tau}]} b$ we start in x_0, jump to x_1 with probability one, and from x_1 it is possible to jump to state x_a. Note that in x_a, the probability of jumping to x_1 is 1. Hence, the probability of the cycle (x_1, x_a, x_1) does not vanish asymptotically (in fact, it converges to one). However, the exit rate of state x_a equals λ instead of $\lambda + \mu$, so the probability of leaving x_a before the time bound $\bar{\tau}$ has passed goes to zero. Hence, both paths with cycles in the interior and with cycles that go through x_a have vanishing probability, so the straight path is the only dominant path.

As is obvious from (3.2), multiplying $\bar{\tau}$ by a constant is the same as multiplying λ and μ by the same constant. The regime $\bar{\tau} \downarrow 0$ (short mission time) is hence the same regime as previously, with the exception that μ decreases by the same rate. The result is that the probability of cycles does *not* vanish asymptotically. However, the exit rate of not just the initial state but that of *all* states decreases. Hence, in the limit of $\bar{\tau} \downarrow 0$ the path with the smallest number of steps will be dominant. The unique path with the smallest number of steps is the straight path.

Note that when the exit rate of a state vanishes asymptotically, drawing the sojourn time in this state using importance sampling (e.g., using forcing as described in Section 1.3.4) can lead to large gains in terms of estimator performance. We will discuss forcing techniques in greater detail in Sections 3.2.2 and 3.2.4.

3.2.2 The Regime $\mu \to \infty$ (Fast Component Repairs)

The second regime that we discuss is $\mu \to \infty$ (fast component repairs), which will be the focus of Chapter 4 (albeit for networks of parallel birth-death processes). For the conditional unreliability, it holds that this regime is completely equivalent to $\lambda \downarrow 0$; after all, the gambler's ruin probabilities as given in (3.5) depend only on the ratio $\frac{\mu}{\lambda}$. Hence, the analysis of the previous section holds and the straight path is the only dominant path. For the unreliability of $\neg a \, \mathsf{U}^{[0,\bar{\tau}]} b$ we observe that the exit rates of the states in the '*interior*' do not converge to μ, as is the case for $\lambda \downarrow 0$, but to ∞. Hence, probability of completing each finite path before $\bar{\tau}$ goes to one, meaning that the time aspect of this setting ceases to be important. As a result, this setting is the same as the setting of the conditional unreliability discussed previously.

In the setting of $\Diamond^{[0,\bar{\tau}]} b$, we again have that the probability of cycles in the interior vanishes, but the probability of the cycle (x_1, x_0, x_1) does *not*. Furthermore, the exit rates in x_0 and x_1 do not go to zero. Hence, starting from x_0, the path that jumps to x_1, falls back to x_0 and then takes the straight path to x_n does not vanish asymptotically, and is by implication part of the set of dominant paths. This will be discussed further in Chapter 4.

3.2.3 The Regime $n \to \infty$ (Many Spares)

The regime $n \to \infty$ (many spares) is the odd one out in this section as the structure of the relevant state space changes as the parametric setting progresses towards the extreme values associated with the regime. Hence, the straight path from x_0 to $\vec{x}_n$ is *not* the same for different values of n. While this regime can be interpreted in a reliability framework as the situation in which system failure is rare because the number of spare components is overwhelming, it is more common to see this regime analysed in a biological, chemical or queueing context. There exists vast literature on results for this regime based on, e.g., large deviations theory and fluid limits. While we give some elementary results concerning the dominant paths in this setting, this regime has no further focus in this thesis. We do note that this regime is very challenging from a computational point of view, as the state space explosion problem becomes worse in this setting.

For the conditional reliability, it holds that the gambler's ruin probabilities of (3.5) grow ever closer to $(\frac{\lambda}{\mu})^{n-i}$ as n goes to infinity. The straight path, with probability $(\frac{\lambda}{\lambda+\mu})^{n-i}$, does not come to dominate the true probability by itself; in fact, its relative contribution vanishes in the limit. Substituting $(\frac{\lambda}{\mu})^{n-i}$ for $w(x_i)$ into (1.13) instead of the straight path probability, we obtain a change of measure in which λ and μ are interchanged. This change of measure is well-known in the literature [84] and can be shown to have several nice properties, including BRE (see Section 1.3.3). One especially nice property of this change of measure is the fact that the likelihood ratio resulting from a cycle equals 1, so that a single run of the simulation procedure either returns 0 or a constant, namely $(\frac{\lambda}{\mu})^{n-i}$ (see also [62]).

For the unreliability, it holds that as n increases, it becomes increasingly unlikely

to complete even the straight path before the time bound $\bar{\tau}$. We state here as a conjecture that in this setting, the straight path by itself again comes to dominate the probability of interest. The reason is that as n increases, the relative likelihood of completing $n+1$ exponentially distributed time steps compared to the completion of n exponentially distributed time steps goes to zero. While this may be interesting from a theoretical point of view, we are unaware of case studies in which the rarity of the event of interest stems from the fact that there are so many spares that is unlikely that these spares all fail within the mission time.

3.2.4 The Regime $\bar{\tau} \to \infty$ (Large Mission Time)

As mentioned in Section 3.1.2, the regime $\bar{\tau} \to \infty$ is the only one for which we consider the property $\neg b \,\mathsf{U}^{(\bar{\tau},\infty)} b$ rather than $\neg a \,\mathsf{U}^{[0,\bar{\tau}]} b$ and/or $\Diamond^{[0,\bar{\tau}]} b$. The motivation for this regime comes from Markov reward models where the event of interest is "collecting *sufficient reward* before absorption". This problem can be shown to be equivalent [5] to the problem of estimating $\mathbb{P}(\neg b \,\mathsf{U}^{(\bar{\tau},\infty)} b)$.

When the amount of time that needs to be spent before the goal set is reached increases, it becomes increasingly likely that this rare event occurs when states that have a very low exit rate are visited often. In Section 3.3, we argue that, given a timeless execution path ω, when $\bar{\tau}$ increases, the fraction of time spent in the state with the lowest exit rate in ω will go to one — assuming this state is unique. If there are k states with the lowest exit rate in a path, then it seems reasonable that, as $\bar{\tau}$ increases, $\bar{\tau}/k$ time units will come be spent in each of these states. In a birth-death process, the state with the lowest exit rate is x_a, so taking the loop (x_1, x_a, x_1) will become increasingly attractive as $\bar{\tau}$ increases. However, it is unclear what the dominant paths are in this situation (if they even exist). We state here as a conjecture that the optimal change of measure for drawing sample paths in this setting will not be Markovian; that it will initially take the loop (x_1, x_a, x_1) with high-probability but that the probability of taking this loop decreases each time it is taken.

While the problem of drawing sample paths is certainly hard, the problem of drawing sample times given a sample path is already non-trivial, as we will see in Section 3.3 (which is the only other section in which the regime $\bar{\tau} \to \infty$ and the probability $\mathbb{P}(\neg b \,\mathsf{U}^{(\bar{\tau},\infty)} b)$ will be discussed).

3.3 The Regime $\bar{\tau} \to \infty$: Fast Simulation for Slow Paths

In this section, we zoom in on efficient simulation for the regime $\bar{\tau} \to \infty$. As we argued in Section 3.2.4, the dominant paths for the regime $\bar{\tau} \to \infty$ in birth-death processes have no easy shape and need not even exist — hence, basing a change of measure on the dominant paths is impossible. However, *given* a good change of measure for the timeless paths, it is possible to construct a well-performing change of measure for the sojourn times in the states of the drawn sample paths. In fact, for estimating the probability $\mathbb{P}(\neg b \,\mathsf{U}^{(\bar{\tau},\infty)} b)$ in *any* CTMC, we can use the following *two-step* approach: in the first step of each simulation run, the simulator samples a

timeless path through the state space, and in the second step it samples the sojourn times for that path.

In this section, we specifically present work concerning the second subproblem: to estimate, given a sample path, the probability that it takes more than $\bar{\tau}$ time units to complete this path. We derive an efficient importance sampling simulation algorithm for it, drawing sojourn times from a distribution that closely resembles the conditional distribution given the rare event of interest.

In Section 3.3.1, we study the conditional distribution of the sojourn times. In Section 3.3.2, we describe our simulation algorithm. In Section 3.3.3, we provide some numerical experiments.

3.3.1 Conditional Sojourn Times

As noted above, we assume that a path ω through a CTMC is already given, consisting of n states $x_1, \ldots, x_n$; only the sojourn times in the states on this path are unknown, but the rates of the states are given as $\lambda_1, \ldots, \lambda_n$, some of which may be identical. As mentioned earlier, this path can be seen as a pure birth process, as depicted in Figure 3.2. We now proceed to analyze the behaviour of the sojourn times T_j in the individual states j of this path, conditional on absorption occurring *after* some time bound $\bar{\tau}$. The results of this section will be used later to obtain an efficient simulation algorithm.

The probability density of the sojourn time T_j is given by $f_j(x) = \lambda_j e^{-\lambda_j x}$, but we are interested in the distribution of T_j *conditional* on occurrence of the event $T > \bar{\tau}$, where $T \triangleq \sum_{j=1}^{n} T_j$. Considering without loss of generality $j = 1$, we condition on the value of T_1 to find

$$\mathbb{P}(T_1 > t | T > \bar{\tau}) = \int_t^\infty \frac{f_1(t_1)}{\mathbb{P}\left(T > \bar{\tau}\right)} \mathbb{P}\left(T - T_1 > \bar{\tau} - t_1\right) dt_1,$$

hence

$$f_1(t|T > \bar{\tau}) = \begin{cases} \dfrac{f_1(t)}{\mathbb{P}\left(T > \bar{\tau}\right)} \mathbb{P}(T - T_1 > \bar{\tau} - t) & \text{if } t < \bar{\tau}, \\ \dfrac{f_1(t)}{\mathbb{P}\left(T > \bar{\tau}\right)} & \text{otherwise.} \end{cases} \tag{3.7}$$

This expression contains the probability $\mathbb{P}\left(T > \bar{\tau}\right)$ which we are trying to estimate. Therefore our goal is now to obtain insight into the behaviour of $f_1(t|T > \bar{\tau})$ for large $\bar{\tau}$ so we can construct a good approximation for $\mathbb{P}(T > \bar{\tau})$ in the next section.

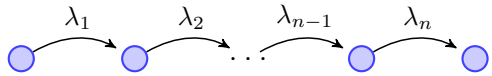


Figure 3.2: Path ω, seen as a pure birth process.

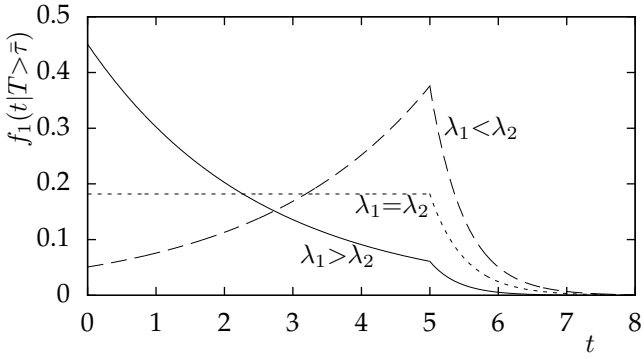


Figure 3.3: $f_1(t|T > \bar{\tau})$ for different parameter values λ_1 and λ_2, with $\bar{\tau} = 5$. Solid line: $\lambda_1 = 2.4, \lambda_2 = 2$. Dotted line: $\lambda_1 = 2.2, \lambda_2 = 2.2$. Dashed line: $\lambda_1 = 2, \lambda_2 = 2.4$.

We start by making (3.7) explicit for a two-state path $\omega = (x_1, x_2)$ with rates λ_1 and λ_2, $\lambda_1 \neq \lambda_2$. Then

$$f_1(t|T > \bar{\tau}) = \begin{cases} \dfrac{\lambda_1 e^{-(\lambda_1-\lambda_2)t}}{\frac{\lambda_1}{\lambda_1-\lambda_2} + \frac{\lambda_2}{\lambda_2-\lambda_1} e^{-(\lambda_1-\lambda_2)\bar{\tau}}} & \text{if } t < \bar{\tau}, \\ \dfrac{\lambda_1 e^{-\lambda_1 t}}{\frac{\lambda_1}{\lambda_1-\lambda_2} e^{-\lambda_2\bar{\tau}} + \frac{\lambda_2}{\lambda_2-\lambda_1} e^{-\lambda_1\bar{\tau}}} & \text{otherwise.} \end{cases} \tag{3.8}$$

Note that the same expression holds for $f_2(t|T > \bar{\tau})$ after interchanging λ_1 and λ_2.

The shape of this function for $t > \bar{\tau}$ is always exponential with rate λ_1. However, the shape of the part where $t < \bar{\tau}$ depends on the parameter setting, where we distinguish three cases. For $\lambda_1 > \lambda_2$, this part is still *negative* exponential albeit with a different parameter, namely $\lambda_1 - \lambda_2$. However, for $\lambda_1 < \lambda_2$, this part is *positive* exponential, again with parameter $\lambda_1 - \lambda_2$. In between, as λ_1 and λ_2 become equal, this part approaches a constant. This is illustrated in Figure 3.3. With the same parameters, in Figure 3.4, the time bound $\bar{\tau}$ was increased sixfold, illustrating the limit behaviour of the system for large $\bar{\tau}$. For $\lambda_1 > \lambda_2$ we see that the probability mass right of $\bar{\tau}$ vanishes, so we can approximate the function (3.8) by a simple exponential density with rate $\lambda_1 - \lambda_2$.

It is also interesting to observe how the *expected share* of the burden of consuming $\bar{\tau}$ time units is distributed over the states. One easily derives the following from (3.8):

$$\mathbb{E}(T_1|T > \bar{\tau}) \sim \begin{cases} \bar{\tau} & \text{if } \lambda_1 < \lambda_2 \\ \bar{\tau}/2 & \text{if } \lambda_1 = \lambda_2 \\ (\lambda_2 - \lambda_1)^{-1} & \text{if } \lambda_1 > \lambda_2, \end{cases}$$

with $\sim$ meaning that the ratio of left- and right-hand side goes to 1 as $\bar{\tau} \to \infty$. This is illustrated in Figure 3.5. We see that when the rates differ, almost all of the time $\bar{\tau}$ is typically spent in the state with the lowest rate, while the time spent in the other state tends to a constant.

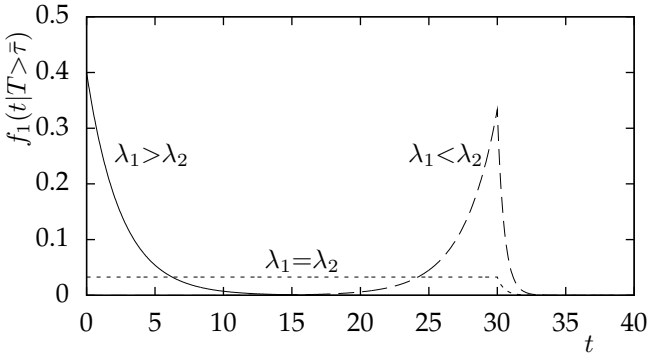


Figure 3.4: Same choices for the values of the parameters λ_1 and λ_2 as in Figure 3.3, but with $\bar{\tau} = 30$.

It seems reasonable to assume that these core observations do not just hold for two-state paths but for any path ω. Denote the lowest rate by β_1 and the second-lowest by β_2, and let r_i be the number of times rate β_i occurs on the path. Then, in the limit for large $\bar{\tau}$, a state i whose rate $\lambda_i \neq \beta_1$ will contribute only an exponentially distributed amount of time with the *bounded* mean $(\lambda_i - \beta_1)^{-1}$. If $r_1 = 1$, then the single state with rate β_1 will account for an amount that has an asymmetric Laplace distribution peaking at $t = \bar{\tau}$ with rates $\beta_2 - \beta_1$ on the left side and β_1 on the right side. If there are $r_1 > 1$ states with rate β_1, then the expected contribution of each of these states is $\bar{\tau}/r_1$, and the conditional sojourn time in each state has an exponential distribution with rate β_1 to the right of $\bar{\tau}$, but a *polynomial* density with degree $r_1 - 2$ left of $\bar{\tau}$.

3.3.2 Simulation

We now proceed to construct an efficient simulation estimator for our probability of interest, namely $\pi = \mathbb{P}(T > \bar{\tau})$ given a path ω. The standard simulation estimator

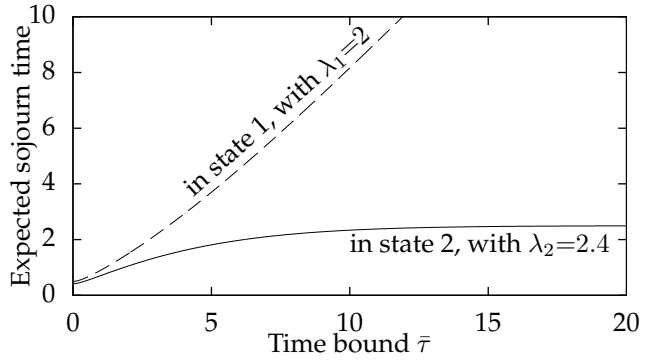


Figure 3.5: Expected sojourn times as a function of $\bar{\tau}$.

for π is

$$\hat{\pi} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{\sum_j t_{ij} > \bar{\tau}}, \tag{3.9}$$

where t_{ij} is the sampled sojourn time (with density $f_j(t)$) for state j in the ith simulation run, N is the number of simulation runs, and $\mathbf{1}$ the indicator function. Importance sampling in this setting boils down to the following: we draw the samples t_{ij} from a different density $g_j(t)$ and weight the result using the following likelihood ratio:

$$\hat{\pi}^* = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{n} \frac{f_j(t_{ij})}{g_j(t_{ij})} \mathbf{1}_{\sum_j t_{ij} > \bar{\tau}}, \tag{3.10}$$

Since the exact calculation of $f_j(t|T > \bar{\tau})$ is problematic is general, we propose to use the following approximation instead, inspired by the findings in the previous section:

$$g_j(t) = \begin{cases} (\lambda_j - \beta_1) \cdot e^{-(\lambda_j - \beta_1) \cdot t} & \text{if } \lambda_j > \beta_1 \\ r_1/\bar{\tau} \cdot e^{-r_1/\bar{\tau} \cdot t} & \text{if } \lambda_j = \beta_1 \text{ and } r_1 = 1 \\ h(t|\beta_1, \beta_2) & \text{otherwise,} \end{cases} \tag{3.11}$$

with β_i and r_i defined as before, and $h(t|\beta_1, \beta_2)$ is given by the right-hand side of (3.8) with each λ_i replaced by β_i.

In practical applications where the CTMC is not a pure birth process, the above algorithm for each simulation run i should be preceded by a phase in which the path itself (i.e., the set of states) is sampled, possibly also using importance sampling (cf. the two-phase approach discussed in the beginning of the section).

3.3.3 Simulation Results

In this section we empirically demonstrate the effectiveness of the method. We compare standard Monte Carlo (MC) simulation using (1.3) to the new importance sampling (IS) approach using (3.10) and (3.11). In each table, the estimates are given in the column '$\hat{\pi}$' and the bounds of the 95%-confidence interval (CI) are given in the next column. Throughout this section, all results are based on 10^7 simulation runs for MC and 10^5 simulation runs for IS — this is shown in the column 'N' in the tables for emphasis. In all cases, we also include a numerical approximation for π, obtained using the model checking tool PRISM. It is shown in the column 'π' in the tables.

In Table 3.1, we consider a two-state path ω with unequal rates. We see that the method works well; the very slow increase of the confidence interval width as $\bar{\tau}$ increases suggests the relative error is in fact upper-bounded (implying the BRE-property, see Section 1.3.3).

In Table 3.2, we set $\lambda_1 = \lambda_2$. The results are still good but somewhat less accurate, which can be explained by the poor resemblance between $g_1(t)$ and $f_1(t|T > \bar{\tau})$ for $t < \bar{\tau}$.

$\bar{\tau}$	method	$\hat{\pi}$	95%-CI-bounds	N	π (PRISM)
5	MC	$2.541{\cdot}10^{-4}$	$\pm\, 0.099{\cdot}10^{-4}$	10 000 000	$2.4168{\cdot}10^{-4}$
	IS	$2.367{\cdot}10^{-4}$	$\pm\, 0.048{\cdot}10^{-4}$	100 000	
7	MC	$5.3{\cdot}10^{-6}$	$\pm\, 1.4{\cdot}10^{-6}$	10 000 000	$4.7363{\cdot}10^{-6}$
	IS	$4.833{\cdot}10^{-6}$	$\pm\, 0.115{\cdot}10^{-6}$	100 000	
9	MC	$3{\cdot}10^{-7}$	$\pm\, 3{\cdot}10^{-7}$	10 000 000	$8.9299{\cdot}10^{-8}$
	IS	$9.133{\cdot}10^{-8}$	$\pm\, 0.248{\cdot}10^{-8}$	100 000	
100	MC	—	—	10 000 000	$8.3034{\cdot}10^{-87}$
	IS	$8.191{\cdot}10^{-87}$	$\pm\, 0.824{\cdot}10^{-87}$	100 000	

Table 3.1: Simulation results for estimating $\mathbb{P}(T > \bar{\tau})$ in a path as depicted in Figure 3.2 with two states, $\lambda_1 = 2$, $\lambda_2 = 2.4$.

$\bar{\tau}$	method	$\hat{\pi}$	95%-CI-bounds	N	π (PRISM)
5	MC	$2.015{\cdot}10^{-4}$	$\pm\, 0.088{\cdot}10^{-4}$	10 000 000	$2.0042{\cdot}10^{-4}$
	IS	$2.011{\cdot}10^{-4}$	$\pm\, 0.037{\cdot}10^{-4}$	100 000	
7	MC	$2.8{\cdot}10^{-6}$	$\pm\, 1.0{\cdot}10^{-6}$	10 000 000	$3.3629{\cdot}10^{-6}$
	IS	$3.347{\cdot}10^{-6}$	$\pm\, 0.075{\cdot}10^{-6}$	100 000	
100	MC	—	—	10 000 000	$6.3039{\cdot}10^{-94}$
	IS	$6.638{\cdot}10^{-94}$	$\pm\, 0.578{\cdot}10^{-94}$	100 000	

Table 3.2: Simulation results for estimating $\mathbb{P}(T > \bar{\tau})$ in a path as depicted in Figure 3.2 with two states, $\lambda_1 = \lambda_2 = 2.2$.

Finally, in Table 3.3 we show results for a path with 50 states and 25 different rates. Note that direct (analytical) calculation of the true probability is not numerically feasible in this case, so we have to resort to either iterative numerical techniques such as the ones implemented in PRISM or simulation-based methods.

$\bar{\tau}$	method	$\hat{\pi}$	95%-CI-bounds	N	π (PRISM)
12	MC	$2.099{\cdot}10^{-2}$	$\pm\, 0.009{\cdot}10^{-2}$	10 000 000	$2.0282{\cdot}10^{-2}$
	IS	$2.054{\cdot}10^{-2}$	$\pm\, 0.033{\cdot}10^{-2}$	100 000	
24	MC	$7{\cdot}10^{-7}$	$\pm\, 5{\cdot}10^{-7}$	10 000 000	$3.9616{\cdot}10^{-7}$
	IS	$4.000{\cdot}10^{-7}$	$\pm\, 0.099{\cdot}10^{-7}$	100 000	
100	MC	—	—	10 000 000	$2.0867{\cdot}10^{-39}$
	IS	$2.104{\cdot}10^{-39}$	$\pm\, 0.119{\cdot}10^{-39}$	100 000	

Table 3.3: Simulation results for estimating $\mathbb{P}(T > \bar{\tau})$ in a path as depicted in Figure 3.2 with fifty states, $\lambda_i = \lfloor \frac{i+1}{2} \rfloor$, $i = 1, \ldots, 50$.

Having demonstrated that the method works well for the pure-birth processes for which it was intended (as the second step of the two-step approach), we also give an example *derived from* a Markov-reward model involving an M/M/5/5 queue. The resulting Markov chain is a birth-death process with 6 states labeled $0, \ldots, 5$, with birth rates $0.1 \cdot e^{5-k}$ in state $k = 0, \ldots, 4$, and death rates $k \cdot e^{5-k}$ in state

$k = 1, \ldots, 5$. We start in 5 and the absorbing state is 0.

Generating appropriate sample paths using standard simulation, and then drawing sojourn times with the algorithm from this section, leads to the results reported in Table 3.4. These results look promising, with a relative error growing just linearly in $\bar{\tau}$. For larger $\bar{\tau}$, however, generating sample paths becomes more difficult, and importance sampling will be needed here as well.

$\bar{\tau}$	method	$\hat{\pi}$	95%-CI-bounds	N	π (PRISM)
1	MC	$2.455{\cdot}10^{-2}$	$\pm\, 0.010{\cdot}10^{-2}$	10 000 000	$2.4599{\cdot}10^{-2}$
	IS	$2.467{\cdot}10^{-2}$	$\pm\, 0.042{\cdot}10^{-2}$	100 000	
2	MC	$2.193{\cdot}10^{-4}$	$\pm\, 0.092{\cdot}10^{-4}$	10 000 000	$2.0927{\cdot}10^{-4}$
	IS	$2.161{\cdot}10^{-4}$	$\pm\, 0.095{\cdot}10^{-4}$	100 000	
4	MC	—	—	10 000 000	$1.5064{\cdot}10^{-8}$
	IS	$1.453{\cdot}10^{-8}$	$\pm\, 0.167{\cdot}10^{-8}$	100 000	

Table 3.4: Simulation results for the M/M/5/5 queue.

3.4 Conclusions

In this chapter, we have given an overview of common performance measures and rarity regimes. Of particular interest are the regimes $\mu \to \infty$, which will be the subject of Chapter 4, and the regime $\lambda \downarrow 0$, which will be the subject of Chapters 5 and 7. For these regimes we have described the typical shape of the dominant paths in a birth-death process; a crucial aspect of our approach in the next chapters will be the search for these dominant paths. We also touched upon the regime $n \to \infty$ and presented some basic results from the literature for this regime.

For the regime $\bar{\tau} \to \infty$, we have presented explicit results and useful approximations for the conditional distribution of sojourn times on a given path in a Markov chain, given that their sum exceeds $\bar{\tau}$. The resulting expressions are relatively simple and yield insight into how this rare event typically happens. Based on these insights we have constructed an importance sampling change of measure and shown that it performs well. A topic of future research could be to identify the dominant paths in the regime $\bar{\tau} \to \infty$ in general Markov chains, after which we can apply the method presented here to those paths. It will also be interesting to consider (rare) events in which both a time- and reward-bound play a role.

CHAPTER 4

Networks of Parallel Birth-Death Processes

In the previous chapter we discussed the single birth-death process, which in a reliability setting typically corresponds to a computer system consisting of components of a single type — the births and deaths in the model then represent failures and repairs of components respectively. Of course, the assumption that all components in a computer systems share the same properties is not very realistic. Hence, the focus of this chapter will be on *networks* of parallel birth-death processes, a model class that corresponds to computer systems consisting of several independent component types. This setting includes several well-known case studies from the reliability engineering literature such as the Distributed Database System (DDS), which, despite being studied since (at least) the late seventies [98], remains a relevant model for modern processor and data storage systems. We are mainly interested in the '*fast component repairs*' regime of Section 3.2.2, but we will also consider the '*slow component failures*' regime of Section 3.2.1. We will consider both the (transient) unreliability and the (steady-state) unavailability. Unlike Section 3.2, we do not impose that the component failure and repair rates remain constant when one or several components have already failed. We will also consider several different repair strategies.

The outline of this chapter is as follows. In Section 4.1, we discuss the strategy of *standard dedicated repair*: each component type has a dedicated repairmen who begins work immediately when a component of his type fails. In Section 4.2, we discuss the repair strategies of *deferred repair*: in this setting we still have a dedicated repairman, but repair is only begun when several components of a single type have failed. In Section 4.3 we apply our techniques to the even more complicated setting in which the components share a single repairman, which repairs components using First Come First Serve (FCFS). Finally, we conclude the chapter in Section 4.4.

4.1 Standard Dedicated Repair

The focus of this section will be on multicomponent systems with a standard dedicated repair strategy. The section is structured as follows. In Section 4.1.1 we discuss the model setting, which has a nice structure as an SPN (see Section 1.1.3). In Section 4.1.2 we introduce the DDS — the case study that we consider in this

section. In Section 4.1.3 we propose an efficient importance sampling scheme for networks of birth-death processes such as the DDS. In Section 4.1.4 we empirically demonstrate the good performance of this importance sampling scheme.

4.1.1 Model Description

Remember from Section 3.1.1 that a single birth-death process is a one-dimensional CTMC in which there are at most 2 transitions leaving every state. A (non-trivial) network of parallel birth-death processes is a parallel composition of d CTMCs, $d > 1$, meaning that the state space equals $\mathbb{N}^d$ and that we have at most $2d$ transitions per state, namely λ_j and μ_j, $j = 1, \ldots, d$. Each state is then a vector and is written as $\vec{x}$, and writing $\vec{e}_j$ for the unit vector with 1 as the value in the jth dimension, the λ_j-transition takes the state from $\vec{x}$ to $\vec{x}+\vec{e}_j$, whereas the μ_j-transition takes the state from $\vec{x}$ to $\vec{x} - \vec{e}_j$. In this setting, the taboo set (if it is relevant to have one) is characterised by a single state $\vec{x}_a$, which equals either $\vec{x}_0$ or a duplicate if we require that the system does not start in a taboo state. The goal set is made up of those states that correspond to an overflow in at least one of the d birth-death processes. Formally, this means that for given overflow boundaries n_j, $n_j \in \mathbb{N}\ \forall j \in \{1, \ldots, d\}$, the goal set is the union of all sets in which there is a $j \in \{1, \ldots, d\}$ such that the jth element of $\vec{x}$ is at least n_j. We allow some rates to become zero; this can be realistic as, e.g., in a reliability setting the failure rate of components of a certain type may be zero when all components are broken. However, we assume for all $\vec{x}$ not in the goal set that there is positive probability of the jth element of $\vec{x}$ reaching n_j, i.e., it is possible from each state to reach overflow in each of the d birth-death processes.

Another way to view networks of parallel birth-death processes is as instances of stochastic Petri nets (SPNs; see Section 1.1.3). In Figure 4.1 a network of 9 parallel birth-death processes (e.g., the DDS of Section 4.1.2) is depicted as an SPN. The component types are represented by places, one for the active components and one for the broken components, and the transitions represent component failures and repairs. For component type j, $j = 1, \ldots, d$, the total amount of tokens in the two places is always the same — this is assumed to be a parameter inherent to the model and at most equal to n_j. Hence, for each component type the number of tokens in one place is then completely determined by the number of tokens in the other. The state of the system can therefore be represented by a d-dimensional vector $\vec{x} = (x_1, \ldots, x_d)$ such that the x_j corresponds to the number of *broken* components of type j.

4.1.2 The Distributed Database System (DDS)

In this section we formally describe the DDS. We first describe the interpretation of the individual birth-death processes that make up the network. Next, we describe the meaning of system failure in this model setting. Finally, we discuss the model parameters and discuss how they can be generalised.

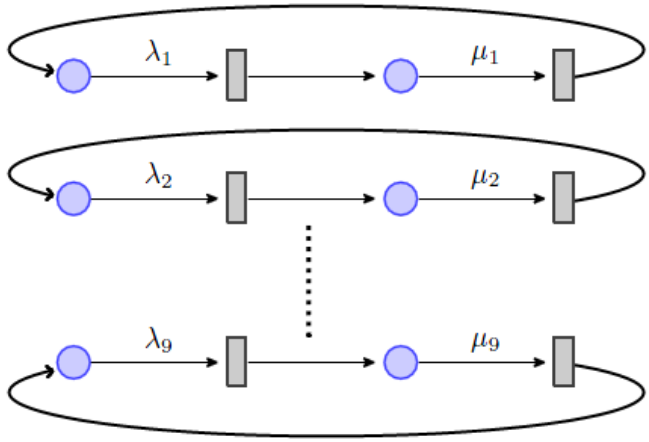


Figure 4.1: A network of 9 parallel birth-death processes, modelled as an SPN. For each individual birth-death process, the number of tokens in the left place corresponds to the number of functioning components, and the second place to the broken components.

Model Description

The distributed database system is a benchmark problem in the field of dependability evaluation [102]. It was recently studied in [16], and a variant was studied in [43]. We study the DDS in this section because it can be seen as part of a more general class of systems consisting of parallel component types. We assume that the system as a whole is fault-tolerant, and that the probability of system failure is low either because of the component failure rates being low or because of the repair rates being high.

Specifically, the distributed database system consists of 24 disks that are grouped together in 6 clusters of 4 disks, 4 disk controller units divided into two sets that each access three disk clusters and a processor that accesses the disk controllers. The processor has a spare that takes over in case of failure. There is one repair facility for each of the six disk clusters, one for each of the two sets of disk controllers and one for the processor and its spare. The system is depicted in Figure 4.2.

We can distinguish 9 component *types*. Types $i = 1, \ldots, 6$ represent the disks in cluster i, types 7 and 8 represent the disk controllers in sets 1 and 2 respectively and type 9 represents the processors. The interfailure times and repair durations are assumed to be exponentially distributed. Let $\vec{x}$ be a vector in $\mathbb{N}^9$ in which each element x_i denotes how many components of type i have failed. We call this vector the *state* of the process. The system will be assumed to start in an initial state $\vec{x}_0$ at time $t_0 = 0$.

Let $\mathcal{D} = \{1, \ldots, 9\}$ be the set of component types. The failure and repair rates of

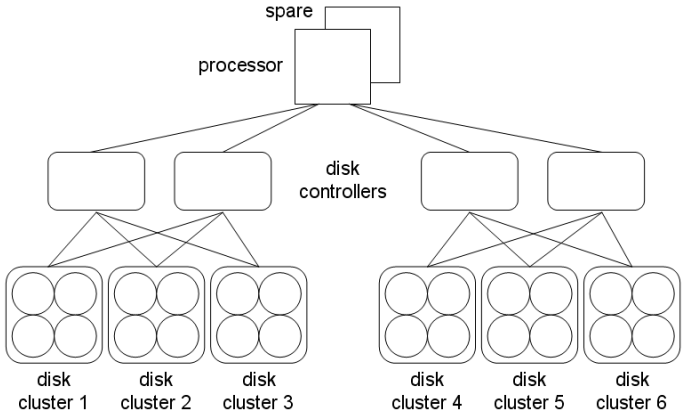


Figure 4.2: A distributed database system

components of these types may depend on the current state, so let the failure rates be some nonnegative function $\lambda_i(\vec{x})$ for each component type $i \in \mathcal{D}$ and $\vec{x} \in \mathbb{N}^9$. Let the repair rates similarly be given by nonnegative functions $\mu_i(\vec{x})$. The failures and repairs are called *transitions*. The repair rate of component type i can only be positive when there is at least one failed component of type i, and the failure rate can only be positive if there are components of type i left that are operational.

Operation and Failure

The system is said to be operational if a processor can access all the data in the disks — this condition is satisfied if each of the following subconditions holds:

1. at least one processor is up,
2. at least one disk controller in each of the controller clusters is up, and
3. at least three disks are up in each of the six disk clusters.

As mentioned in Section 4.1.1, we assume in general that system failures occur if for at least one component type i, a specified number n_i of components has failed. System failure in the benchmark setting falls into this category. A state in the system satisfies the atomic property b if and only if the system is failed in this state.

Using this definition of system failure we can formalise what kind of measures we will estimate. By the unreliability we mean $\pi_{\psi^{\bar{\tau}}}(x_0) = \mathbb{P}_{x_0}(\Diamond^{[0,\bar{\tau}]} b)$, the probability that the system stops being operational at some point before a specified time bound $\bar{\tau}$ as described in Section 3.1.2, starting in x_0. The unavailability is simply the steady-state probability that at some time point t in steady-state the system is not operational.

The Benchmark Case and Generalisations

The failure and repair rates of the individual components are given in Table 4.1. In

	failure rate	repair rate
disks	λ	μ
disk controllers	3λ	μ
processors	3λ	μ

Table 4.1: Failure and repair rates in the benchmark model

the benchmark case (see [16]) we have $\lambda = 1/6000$ and $\mu = 1$. The rates in the literature are per hour, and the time bound $\bar{\tau}$ for the unreliability is 5 weeks, so equal to 840 time units in this setting. The individual components all have the same failure distribution regardless of how many other components are up. E.g., the total failure rate $\lambda_i(\cdot)$ of type 1 components (the first disk cluster) is 4λ when no components are down, 3λ when one component has failed, and so on. The component repair rate of each i is always μ if $x_i > 0$, because there is only repair facility per type.

We also parameterise the number of spares. We introduce a new parameter n and assume there are n processors, n disk controllers per set and $2n$ disks per cluster. For $n = 2$ we are back in the benchmark case. Let failure in this more general setting be defined to occur when either (1) no processor is up, (2) in one disk controller set, no disk controller is up or (3) in one disk cluster at least n disks are down.

4.1.3 Approximating $\pi_{\psi^{\bar{\tau}}}$ using Straight Paths

In this section, we seek to obtain a good approximation w for $\pi_{\psi^{\bar{\tau}}}$ to be substituted into (1.12) to produce a good simulation measure. As mentioned in Section 3.1.3, an obvious candidate for w is the probability of the straight paths to failure: those paths that end in a system failure state and which contain only failure transitions of a single component type. Let d be the number of types. From each state $\vec{x}$, we have d straight paths to failure, one for each type k. Let ν_k failures remain until the critical level n_k is reached for type k, and let the vector of states that are seen along this path be denoted by $[\vec{x}_{k,0}, \ldots, \vec{x}_{k,\nu_k}]$, with $\vec{x}_{k,0}$ equal to $\vec{x}$. The probability of this path being taken equals

$$\mathbb{P}([\vec{x}_{k,0}, \ldots, \vec{x}_{k,\nu_k}]) = \prod_{i=0}^{\nu_k - 1} \frac{\lambda_k(\vec{x}_{k,i})}{\eta(\vec{x}_{k,i})},$$

where $\eta(\vec{x})$ is the exit rate in state $\vec{x}$. We can then use

$$w^*(\vec{x}) = \sum_{k=1}^{d} \prod_{i=0}^{\nu_k - 1} \frac{\lambda_k(\vec{x}_{k,i})}{\eta(\vec{x}_{k,i})} \tag{4.1}$$

as a time-independent approximation of w. From now on, the $*$ in w^* indicates that we only use the straight paths as an approximation.

Unfortunately, some further investigation reveals that the approximation (4.1) is too crude. One shortcoming of w^* is that the most likely path from a state $\vec{x}'$ to system failure might not be one of the d straight paths. In many cases, the most likely path is the path in which the system first returns to state $\vec{x}_a$ and then takes one of the straight paths that determine $w^*(\vec{x}_a)$. This can be seen in Figure 4.3, which depicts the state space of a simplified model with only two component types. Starting from state $\vec{x}'$ in Figure 4.3, the dashed line path (the one going '*north*') is, with realistic rates, much less likely to occur than the solid line path because the former contains one more step where a failure transition needs to win the race from a repair transition.

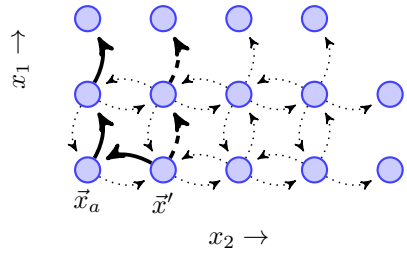


Figure 4.3: A model consisting of two types of components. For one type there are many more spares than the other, but its components fail more quickly.

Accordingly, we also consider the straight paths from state $\vec{x}_a$ for our approximation $w(\vec{x}')$. From state $\vec{x}'$, the system returns to state $\vec{x}_a$ with probability almost equal to one. Therefore, for each state $\vec{x}$ we can use the sum of $w^*(\vec{x})$ and $w^*(\vec{x}_a)$ instead of just the former. As a consequence, the jump from state $\vec{x}_a$ to state $\vec{x}'$ will more often be taken under the new distribution. This is desirable — paths that contain cycles between state $\vec{x}_a$ and the states where one component has failed are almost equally likely as the straight paths from $\vec{x}_a$, as we explained in Section 3.2.2.

However, the contribution of $w^*(\vec{x}_a)$ needs to be time-dependent — cycles to the taboo state are *only* likely when the taboo state's exit rate $\eta(\vec{x}_a)$ is high enough compared to the remaining time $\bar{\tau} - t$. Otherwise, the extra jumps take too much time, which reduces the likelihood of these paths.

For finding a time-dependent $w(\vec{x}, t)$, a crucial insight is that the time-*in*dependent function $w^*(\vec{x}_a)$ is still a good approximation for the probability of hitting a system failure state *during a busy cycle*. For ease of notation, let $c \triangleq w^*(\vec{x}_a)$ and $\lambda_0 \triangleq \eta(\vec{x}_a)$. When the failure rates are low or the repair rates are high, the duration of the busy cycle is almost completely determined by the time spent in state $\vec{x}_a$. Therefore, the time it takes before we reach a system failure state is the sum of M busy cycle durations D_i where M itself is a random variable. Here, the durations D_i are all independent and exponentially distributed approximately with the rates λ_0, while the number M follows a geometric distribution with approximate success parameter c.

From elementary probability theory we know that this sum follows an exponential distribution with rate $\lambda_0 \cdot c$, hence the probability that this is completed before $\bar{\tau} - t$ time units has approximate probability $1 - \exp(-\lambda_0 \cdot c(\bar{\tau} - t))$.

For small x, $1 - e^{-x}$ approximately equals x. Since c is assumed to be small, we can approximate $w(\vec{x}_a, t)$ using the time-dependent function $c\lambda_0(\bar{\tau} - t)$. This motivates our final approximation,

$$w(\vec{x}, t) = \begin{cases} c \cdot \lambda_0 \cdot (\bar{\tau} - t) & \text{if } \vec{x} = \vec{x}_a, \\ w^*(\vec{x}) + c \cdot \lambda_0 \cdot (\bar{\tau} - t) & \text{otherwise.} \end{cases} \tag{4.2}$$

Using (4.2) in (1.12) keeps the estimator efficient when the rarity of the event of interest is not caused by the low component failure rates but rather the high repair rates. Our numerical results will show that this adaptation is crucial in practical situations. To draw sojourn times, we use forcing as described in Section 1.3.4. We do this in all states, even if it is really necessary only in $\vec{x}_a$.

4.1.4 Simulation Results

In this section, we demonstrate that the change of measure of (4.2) produces good results in practice. We compare our method to a few other well-known techniques. The first of these is the standard Monte Carlo method. A more efficient method is that of forcing combined with balanced failure biasing (BFB, see Section 1.3.4). The third simulation method found in the result tables of this section are the estimates produced with the new method using the approximation in (4.2), abbreviated as Path-IS. Finally, we will compare our method to the numerical methods of the model checking tool PRISM.

When we display the experimental results in a table, we first give the statistical estimates. These are either the standard Monte Carlo estimates as in (1.3) or importance sampling estimates as in (1.9), which will be clear from the context, and are given in the form of a 95%-confidence interval. We write π for the unreliability and v for the unavailability (see Section 1.2.1) — $\hat{\pi}$ and $\hat{v}$ are their respective estimates. To the right of the estimates, we state the number of simulation runs used to produce these estimates. The number of simulation runs for each method was picked such that the computation time was comparable to that of PRISM. In the last row(s), we display the numerical solutions and the number of states in the PRISM [67] and Arcade [16] models (the latter tool uses lumping/bisimulation minimisation to reduce the size of the state space). The exact computation and simulation times are specified in the text.

Experimental Setup

There is a fundamental difference between the numerical approach and the statistical approach in the sense that numerical methods (if they converge) give an almost perfect (depending on the stopping criterion) approximation after some fixed time interval. On the other hand, statistical methods produce confidence intervals that

can be made as narrow as one would like, depending on how much time one is willing to spend. The best way to say something about the applicability of an approach for the user is to look at the wall-clock time.

We used a computer with a 2.8 GHz Intel® Core™ 2 Duo processor (32-bit) and 3 GB of RAM, running Windows XP. All simulations were run with a simple Java program that generated (pseudo-)random numbers using a fast Mersenne twister [75]. We used version 3.3.1 of PRISM.

Unavailability

Of the two measures discussed in this section, the unavailability is the easiest to approximate. Because it considers the system when it is in equilibrium, no information about the transient behaviour of the system is needed. Numerical methods to analytically determine or iteratively approximate it are well-established.

First, we will show in Table 4.2 that our results are consistent with the other tools and the literature, namely [16]. The unavailability in [16] was only given in one significant digit, and the total run time was not specified. When we lower the

	$\hat{v}\ (\cdot 10^{-6})$	# samples
MC	3.677 ± 0.778	388 196
BFB	3.647 ± 0.104	169 484
Path-IS	$\mathbf{3.511 \pm 0.035}$	**79 611**
	$v\ (\cdot 10^{-6})$	# states
PRISM	3.498	421 875
Arcade	3	2 100

Table 4.2: Unavailability ($\hat{v}$) results for the benchmark case. $\lambda = 1/6000, \mu = 1, n = 2$.

component failure rate parameter λ from $1/6000$ to $\frac{1}{6} \cdot 10^{-6}$, the simulation estimates agree with the numerical results obtained using PRISM, with the exception of standard Monte Carlo. This is displayed in Table 4.3. Increasing μ from 1 to 1000

	$\hat{v}\ (\cdot 10^{-12})$	# samples
MC	5.847 ± 11.460	386 538
BFB	3.532 ± 0.105	165 943
Path-IS	$\mathbf{3.521 \pm 0.036}$	**78 179**
	$v\ (\cdot 10^{-12})$	# states
PRISM	3.500	421 875

Table 4.3: Unavailability ($\hat{v}$) results when $\lambda = \frac{1}{6} \cdot 10^{-6}; \mu = 1, n = 2$.

gives us similar results, as depicted in Table 4.4 (note that the unavailability values for $\lambda = \frac{1}{6} \cdot 10^{-6}$ and $\mu = 1000$ are exactly the same. This is not a coincidence, as the solution depends only on the transition rates through the ratio λ/μ). In all these cases PRISM does better than the simulation approaches discussed so far —

	$\hat{v}$ $(\cdot 10^{-12})$	# samples
MC	$0 \pm \infty$	384 418
BFB	3.504 ± 0.102	165 115
Path-IS	$\mathbf{3.465 \pm 0.035}$	**76 923**
	v $(\cdot 10^{-12})$	# states
PRISM	3.500	421 875

Table 4.4: Unavailability ($\hat{v}$) results when $\mu = 1000$; $\lambda = 1/6000, n = 2$.

indeed, for models with small state spaces PRISM's steady-state techniques are to be preferred to simulation, regardless of λ or μ. However, if we increase the spare component parameter n from its benchmark value of 2 to 3, the size of the state space increases about 18-fold, as can be seen in Table 4.5. This causes PRISM's computation time to increase, from about 3.4 seconds for Tables 4.2-4.4 to 113.1 seconds for Table 4.6. When we increase n even further, we hit tougher bound-

n	# states	# non-zeros
2	421 875	5 737 500
3	7 529 536	111 329 568
4	66 430 125	1 027 452 600
5	382 657 176	6 087 727 800
6	1 655 595 487	26 853 394 932

Table 4.5: State space sizes and numbers of non-zero entries in the transition rate matrix of the models built by PRISM for different values of n.

	$\hat{v}$ $(\cdot 10^{-9})$	# samples
MC	7.235 ± 6.135	12 977 468
BFB	5.656 ± 0.151	3 434 986
Path-IS	$\mathbf{5.580 \pm 0.015}$	**1 315 050**
	v $(\cdot 10^{-9})$	# states
PRISM	5.578	7 529 536

Table 4.6: Unavailability ($\hat{v}$) results when $n = 3$; $\mu = 1, \lambda = 1/6000$.

aries on the applicability of numerical methods due to the state space explosion problem. For $n \geq 4$, the amount of memory that our system has available for "*creating* [a] *vector for diagonals*" is insufficient and PRISM terminates without giving a solution (even after adjusting the memory usage maxima in PRISM's settings). For $n = 6$, Path-IS still produces accurate estimates when we set the simulation time to a mere 60 seconds, as can be seen in Table 4.7. BFB underestimates the unavailability, a well-known phenomenon when the change of measure being used is not suitable for the problem [33].

	$\hat{v}$ $(\cdot 10^{-16})$	# samples
MC	$0 \pm \infty$	6 708 624
BFB	0.148 ± 0.225	803 752
Path-IS	$\mathbf{1.173 \pm 0.016}$	**205 654**
	v $(\cdot 10^{-16})$	# states
PRISM	N.A.	1 655 595 487

Table 4.7: Unavailability ($\hat{v}$) results when $n = 6$; $\mu = 1, \lambda = 1/6000$.

Unreliability

The unreliability is (from a theoretical point of view) a more interesting case than the unavailability because, unlike the latter, the former is not known in closed form for the models that we consider [44] — hence, we simply have to use numerical and/or statistical methods. First, note that we have defined the unreliability to refer to the probability of system failure before some time point $\bar{\tau}$ (in this case 840 hours), allowing the repair of components in this time interval. In [16] and [102], component repairs were *not* allowed to occur.

In Table 4.8, we display the results for the benchmark case (no repairs allowed). Because PRISM's numerical evaluation was very quick (0.235 seconds), we gave the statistical methods more time (60 seconds), as the purpose of Table 4.8 is only to show that our results are consistent with the literature even when the repair transitions are disabled. Again, no run time was given for Arcade in [16]. Note that standard Monte Carlo and BFB give the best results in this setting because their simplicity allows them to sample many more runs within the (real) time constraint. Also, the event of interest is not rare in this setting so we are not in a setting in which the IS-methods are expected to perform well. When we allow repairs to occur

	$\hat{\pi}$	# samples
MC	0.5981 ± 0.0003	8 304 940
BFB	0.5976 ± 0.0003	5 116 887
Path-IS	$\mathbf{0.5977 \pm 0.0019}$	**93 526**
	π	# states
PRISM	0.5980	421 875
Arcade	0.5980	2 100

Table 4.8: Unreliability ($\hat{\pi}$) results *without* repair ($\mu = 0$), $n = 2$, $\lambda = 1/6000$.

the unreliability drops to approximately 0.0029. It takes PRISM little more than 30 seconds to compute this probability. This computation time does not depend on λ, as it took a comparable amount of time to generate the results of Table 4.9, where we lowered λ to $\frac{1}{6} \cdot 10^{-6}$.

However, when we increase μ, the time that PRISM needs to produce a solution increases along with it. The applied numerical methods require that the transition rate matrix be uniformised, and the uniformisation rate increases linearly in μ.

PRISM's computation time in turn increases linearly in the product of the uniformisation rate and the mission time (see [49], chapter 15). Because the uniformisation rate is so much higher than the original exit rate of the taboo state, many unnecessary self-loops are taken into account. This can heavily slow down the computation. On the other hand, the accuracy of the Path-IS estimate remains constant as μ increases since the jumps out of the taboo state still occur with the same low rate. A few estimates together with PRISM computation times are given in Figure 4.4. Notice that when $\mu = 100$, PRISM takes over half an hour to produce an approximation, while our simulation method can produce a decent estimate in 10 seconds.

	$\hat{\pi}\ (\cdot 10^{-9})$	# samples
MC	$0 \pm \infty$	18 438 588
BFB	2.936 ± 0.024	1 042 866
Path-IS	$\mathbf{2.937 \pm 0.001}$	**992 231**
	$\pi\ (\cdot 10^{-9})$	# states
PRISM	2.936	421 875

Table 4.9: Unreliability ($\hat{\pi}$) results when $\lambda = \frac{1}{6} \cdot 10^{-6}$; $n = 2$, $\mu = 1$.

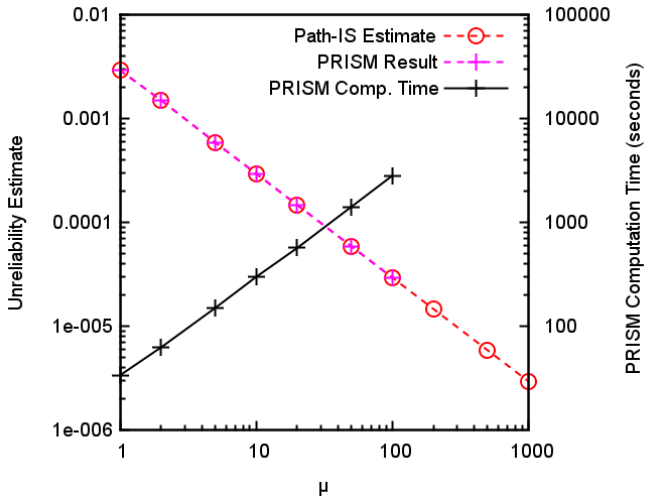


Figure 4.4: Estimates for the probability π (the unreliability) from PRISM (dashed, crosses) and Path-IS (dashed, circles) on left vertical axis; PRISM run time (solid, crosses) on right vertical axis (we did not generate PRISM-results for $\mu > 100$ due to the large computation time). The Path-IS run time was only 10 seconds, but the bounds of the 95%-confidence interval were still not distinguishable from the estimate at this scale.

For high μ (and high $\bar{\tau}$), the confidence intervals of BFB are also noticeably wider

than those of Path-IS. For $\mu = 1000$ (10 second run time), they were 2.943 ± 0.013 and 2.920 ± 0.358 (times 10^{-6}) respectively. The reason is that BFB is not well-suited for the rarity regime of '*fast component repairs*' (the regime discussed in Section 3.2.2).[19]

For higher values of the spare component parameter n, PRISM again starts to suffer from the state-space explosion problem. We omit results for this scenario as they are comparable to the results for the unavailability when n is high.

4.2 General Busy Cycle Durations

Consider again a system that consists of several component types with a dedicated repair unit for each type. We previously assumed that repair would start immediately after a component failure, but now assume that, for one or multiple component types, the cost of having the repairman come over is high. In this setting, it might be more cost-efficient to initiate repair for a component type only after *multiple* components *of this type* have failed. This repair strategy is called *deferred repair*. Generalised versions of BFB were proposed in [60] and [61] for estimating steady-state measures in this model setting.

We cannot trivially assume that the method described in Section 4.1.3 also works well in this new setting; the analysis in this section relies on the busy cycle durations being approximately memoryless, but no matter how small λ/μ is in this new scenario, there will be more than one exponential *'phase'* in which a non-negligible amount of time is spent before system failure. Since μ is very high compared to λ, a non-negligible amount of time is spent *only* in states in which no repair transitions are allowed.

In other words, we need to divide the state space into two subsets: a *typical* set in which repair transitions are not allowed and in which the system spends the bulk of its time ($\vec{x}_a$ in the setting of Section 4.1.3, but a larger number of states in the current setting), and an *atypical* set outside of it. Therefore, the probability distribution of the time until system failure will depend on the phase in the typical region, and phase-type distributions are not memoryless in general. Furthermore a good approximation $w(\vec{x}, t)$ will require that information about a larger set of typical paths is used than just the straight paths of Section 4.1.3. These paths may '*turn*' inside the typical region, and are straight only after entering the atypical region.

In this section we will show how to overcome these problems. In Section 4.2.1 we will describe how the loss of the memoryless property can be overcome, by showing that in an appropriate limiting regime the importance of the current phase will vanish and the time until system failure will again have a memoryless distribution. In Section 4.2.2, we will study the nature of the larger set of relevant paths, and in Section 4.2.3 we will demonstrate the necessity of the analysis in this section by empirically comparing the generalised method (which we shall call Path-IS-G) to the method of Section 4.1.3 (called Path-IS), standard simulation, BFB and the

[19] More refined techniques based on BFB are available for this setting [80], but depart from the standard set-up of importance sampling as discussed in Section 1.3.2.

numerical techniques of PRISM.

4.2.1 Non-Memoryless Busy Cycles

As we will only discuss the problem of non-memorylessness in this section, we consider a very simple model in which the need for extra paths does not occur. This is the case when our system consists of only one component type. A model of such a system is depicted in Figure 4.5, where the squares represent states in the typical set (i.e., where repair transitions are not yet allowed) and the circles represent states in the atypical set.

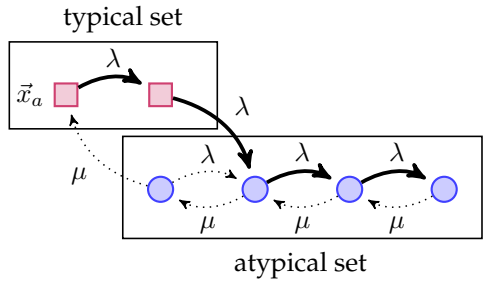


Figure 4.5: A model with a single component type. The repairman starts work only after the second component has failed.

For small λ/μ the duration of a busy cycle will be almost completely determined by the sojourn times in the typical set. Clearly, such a duration approximately has an Erlang(2,λ) distribution, for which the memoryless property does not hold. In more general situations, the distribution of the time spent in the typical set could have any phase-type distribution.

Although the time within a busy cycle is no longer memoryless, the distribution of the number of busy cycles needed before system failure remains geometric, hence, memoryless. We know that λ/μ is small, so c, the probability of system failure during a single busy cycle, will also be small. As a consequence, the expected number of busy cycles before system failure will be large. Interestingly, it can be shown that for small values of c the time until system failure again approximately has an exponential distribution, with rate $c/\mathbb{E}(D)$, with D the duration of a busy cycle as in Section 1.2.1. This is formalised in Theorem 4.1. In this theorem, we prove that despite the fact that the duration of each busy cycle is no longer exponentially distributed, the sum of a geometrically distributed number of these durations is approximately exponentially distributed if the success parameter of the geometric distribution is small. First we will state the theorem, then we explain why we need the theorem, then we prove the theorem and conclude the section with a short discussion on how we use it.

Theorem 4.1. *Let $X_1, X_2, \ldots$ be a sequence of i.i.d. random variables such that $\mathbb{E}(X_1) < \infty$, and let M be a random variable independent of X_i, $i \in \mathbb{N}$, that has a geometric distribution*

with success parameter q. Let $S_n = X_1 + X_2 + \ldots + X_n$. Then

$$\lim_{c \downarrow 0} \mathbb{P}(cS_M > t) = e^{-t/\mathbb{E}(X_1)}.$$

Theorem 4.1 states that for small c, $\mathbb{P}(cS_M > t) \approx \exp(-t/\mathbb{E}(X_1))$, which implies that $\mathbb{P}(S_M > t) \approx \exp(-c \cdot t/\mathbb{E}(X_1))$ which we will use below.

Proof. Since a probability distribution is uniquely characterised by its Laplace-Stieltjes Transform (LST) we will consider the LST of cS_M. For the LST of X_1 we know that, since $\mathbb{E}(X_1) < \infty$, we can write

$$\mathbb{E}\left(e^{-sX_1}\right) = 1 - s\mathbb{E}(X_1) + o(s),$$

where $o(s)$ stands for a function $f(s)$ satisfying $\lim_{s \downarrow 0} \frac{f(s)}{s} = 0$. Furthermore, it is known that the probability generating function (PGF) of a geometrically distributed random variable M with probability c is

$$\mathbb{E}(z^M) = \frac{c}{1 - (1-c)z},$$

Then with M geometrically distributed it holds that

$$\begin{aligned}
\mathbb{E}\left(e^{-scS_M}\right) &= \mathbb{E}\left(\mathbb{E}\left(e^{-scS_M}|M\right)\right) = \mathbb{E}\left(\left[\mathbb{E}\left(e^{-scX_1}\right)\right]^M\right) \\
&= \frac{c}{1 - (1-c)\mathbb{E}\left(e^{-scX_1}\right)} \\
&= \frac{c}{1 - (1-c)(1 - sc\mathbb{E}(X_1) + o(sc))} \\
&= \frac{c}{c + (1-c)sc\mathbb{E}(X_1) + o(sc)} \\
&\xrightarrow{c \downarrow 0} \frac{1}{1 + s\mathbb{E}(X_1)},
\end{aligned}$$

which we recognise as the LST of an exponentially distributed random variable with mean $\mathbb{E}(X_1)$.

□

Using the same linear approximation as in Section 4.1.3 we obtain an approximation $w(\vec{x}, t)$ similar to (4.2) with λ_0 replaced by $1/\mathbb{E}(D)$.

We note here that in the setting in which we will apply the theorem, not all of its assumptions are satisfied: first of all, even if there is only *one* state in the typical set, M and the series $X_1, X_2, \ldots$ are not completely independent. After all, X_M has a different probability distribution than $X_1, X_2, \ldots, X_{M-1}$ (a busy cycle in which system failure occurred went '*deeper*' into the state space than a typical busy cycle, so it can be expected to have lasted longer.) However, we expect this not to have much impact on our conclusions for two reasons: first of all, when c goes to zero, M will grow larger and the relative influence of X_M with respect to $X_1, X_2, \ldots, X_{M-1}$

will vanish. Furthermore, in our setting $c \downarrow 0$ because $\lambda \downarrow 0$ or $\mu \to \infty$, and in exactly these two regimes the difference between the distributions of X_M and the preceding busy cycle durations will vanish as the time needed to go 'deeper' into the state space becomes smaller relative to the time spent in the typical set's only state.

In our situation there is more than one state in the typical set, the point of entry into the typical set will have influence on the next point of entry (it is less likely that this is the same state twice in a row). This means that there is correlation between each X_i and X_j, $i, j > 0$, but this vanishes as $|i - j|$ gets bigger. The assumption that we make is that in our limiting regime, i.e., $c \downarrow 0$, the series of busy cycles until system failure becomes so large that the influence of the correlation between the individual random variables becomes smaller as well. Judging by our empirically determined estimator variances, this assumption is justified.

4.2.2 Non-Rare Paths

The next step is to extend the result of Section 4.2.1 to a system with *multiple* component types — a model of such a system is depicted in Figure 4.6a. It still holds for this system that the duration of the busy cycle is approximately equal to the time spent in the typical set. However, when a system has a large number of component types with comparable failure rates, it is likely that between failures of two components of the same type, a component of another type fails. The '*straight*' paths of Section 4.1.3 do *not* account for such behaviour. As a consequence, the probability c of failure during a busy cycle is underestimated, and the resulting change of measure will be inefficient because the probability of falling back to the origin is made too low.

Another consequence of the added component types is that the expected duration of a busy cycle may be very long. Not only does this impact the efficiency with which we can estimate $\mathbb{E}(D)$, it may also imply that the assumption, made in Section 4.2.1, that the probability of system failure during a busy cycle c is small while $\bar{\tau}$ is relatively large compared to $\mathbb{E}(D)$, is violated for realistic parameter settings.

Hence, we alter the definition of the busy cycle: we now say that a *generalised busy cycle* D' starts and ends when the system jumps from the *atypical* to the *typical* set. Note that the starting times of the busy cycles are no longer i.i.d., but this has a negligible impact. Hence, we can still justifiably approximate the time until system failure by a geometrically distributed number of (generalised) busy cycle durations.

We approximate the probability c of failure in such a generalised busy cycle using the following strategy: we use simulation using the *original* probability distribution (see Section 1.2) to estimate $\mathbb{E}(D')$. This means that we choose a number N_D a priori or let it depend on a bound on the run time. Because a fixed time bound leads to strongly varying accuracies of $\mathbb{E}(D')$ for different n and λ (the number of samples generated per time unit depends strongly on these model parameters) we fixed $N_D = 500$ for the results of Section 4.2.3.

Whenever a new generalised busy cycle starts during this first simulation round,

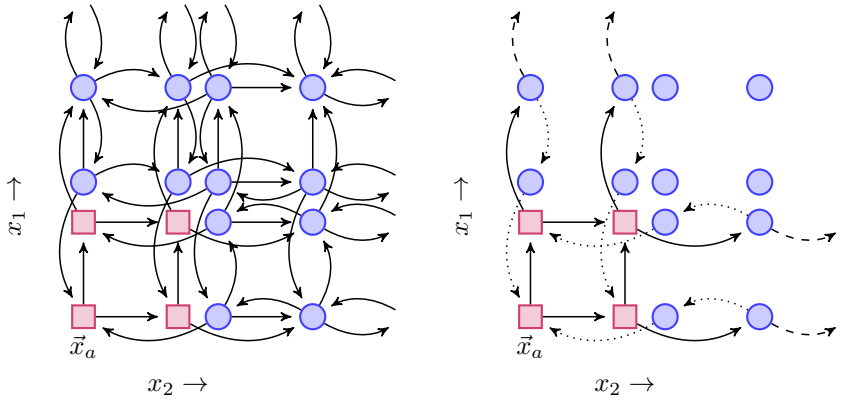


Figure 4.6: A model consisting of two types of components, where for each type, repair starts when two components of that type have failed. The square states represent the typical set, the round states represent the atypical set. In (a) we display all possible transitions; in the square states none of the repairs are yet enabled. In (b) we display only the most important transitions; dashed: transitions in a typical successful busy cycle; dotted: transitions in a typical *un*successful busy cycle; solid: transitions that occur in both.

we record the state $\vec{x}$ in which the typical set is entered. We then find all paths that first move (in any way) through the typical set and then follow a straight path to system failure. In the example of Figure 4.6(b), there would be four of those paths from $\vec{x}_a = (0,0)$ and three from the states $(0,1)$ and $(1,0)$ in the typical set. Then we use the sum of the probabilities of all those paths — we denote this value by $\hat{c}^*(\vec{x})$ — as a single *sample* that is used to find an *estimate* for q. Hence, our approximation for c is a random variable C whose expected value we estimate using the same simulation runs as the ones for $\mathbb{E}(D')$. Finally, after having generated N_D samples, we use the sample means as estimates for $\mathbb{E}(D')$ and $\mathbb{E}(C)$, which leads to our final approximation:

$$w(\vec{x},t) = \begin{cases} \mathbb{E}(C)\cdot(\bar{\tau}-t)/\mathbb{E}(D')\cdot, & \text{if } \vec{x}=\vec{x}_a, \\ \hat{c}^*(\vec{x}) + \mathbb{E}(C)\cdot(\bar{\tau}-t)/\mathbb{E}(D'), & \text{otherwise.} \end{cases} \tag{4.3}$$

In Section 4.2.3 we will only display results for the unreliability, but one can also use the change of measure in (4.3) to estimate the steady-state unavailability. One would then again set $(\bar{\tau}-t)$ to zero and the only change would be the number of paths used for w^*. Note that for the ratio estimator (1.5) we still use busy cycle durations D instead of D' as the latter ones do not form a renewal process.

4.2.3 Simulation Results

In this section, we will empirically compare the simulation distribution based on (4.2) (Path-IS) to the more general simulation distribution based on (4.3) (Path-IS-G). We introduce a vector of repair thresholds $r_i, i \in \{1, \ldots, 9\}$. Repair of component of type i is "*allowed*" when the number of failed components of type i reaches the threshold r_i, and then switched off when all failed components of that type have been repaired. Typically, the addition of these triggers will increase the expected generalised busy cycle duration but also increase the probability q of failure during such a cycle both due to an increased number of ways failure can occur (more paths to failure) and the fact that fewer failure transitions need to '*win the race*' against repair transitions.

Single Component Type

We will first consider the increase in expected busy cycle duration by considering the processor set (which is one of the nine component types) in isolation. We increase $\bar{\tau}$ from the benchmark 840 to $8\,400$ because in the new setting the generalised busy cycle will take so long that the assumption that cycles are completed before system failure is no longer valid. The CTMC that underlies the model will then look like the one in Figure 4.5 — there are r_i states in the typical set and n states in the atypical set. For Figure 4.7, we let $r_i = n/2$ and show the estimates and numerical results of PRISM for increasing n. The probability of interest can be seen to decrease exponentially. In this situation, PRISM is still clearly superior in the sense that its computation times are negligible; the reason is that the model size is very small when only one component type is modelled.

The 95%-confidence intervals in Figure 4.7 are too narrow to be visible. However, to get an idea of how wide they are compared to each other, we display the relative error $\hat{\sigma}/(\hat{\pi}\sqrt{N})$ in Figure 4.8. At $n = 2$, the two methods are still the same as repair starts immediately after a component has failed. However, for larger values of n the generalised approximation Path-IS-G clearly outperforms the original Path-IS. When n (and, hence, all r_i) grows higher, the difference between the two '*estimates*' for $\mathbb{E}(D)$ and $\mathbb{E}(D')$ increases. As a side note: for higher $\bar{\tau}$ the efficiency of the change of measure based on (4.2) will decrease further.

Multiple Component Types

Whereas PRISM is still the most efficient tool for analysing small models such as the Distributed Database System (DDS) with only one component type, its performance is much worse for models with larger values of n. For $n = 3$ and $r_i = 2$ for all i, the size of the state space is $32\,768\,000$ compared to the $7\,529\,536$ of Table 4.5 — the number of non-zero entries in the transition rate matrix increases by a similar factor. PRISM runs out of memory in this situation, so in order to be able to compare the methods we remove two disk sets from the model, resulting in seven component types. This model only has $512\,000$ states and $5\,478\,400$ transitions. It takes PRISM

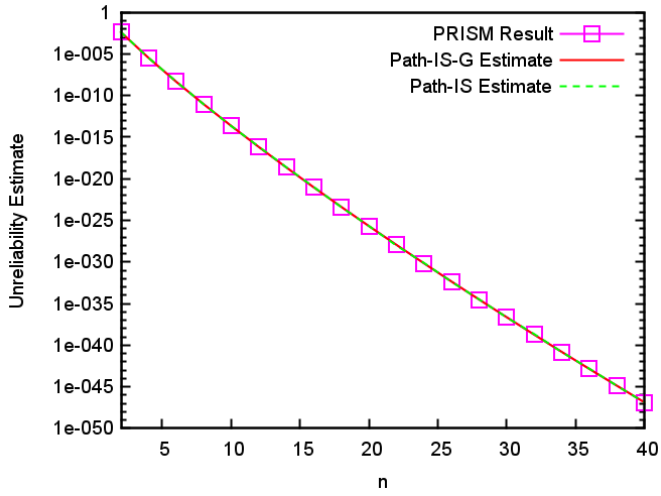


Figure 4.7: Unreliability ($\hat{\pi}$) results for the model with one component type. $\lambda = 1/6000$, $\mu = 1$, $\bar{\tau} = 8\,400$, $r_1 = n/2$. $N = 1000$ samples were drawn for each simulation — the run time for each simulation was less than five seconds. PRISM run time was negligible.

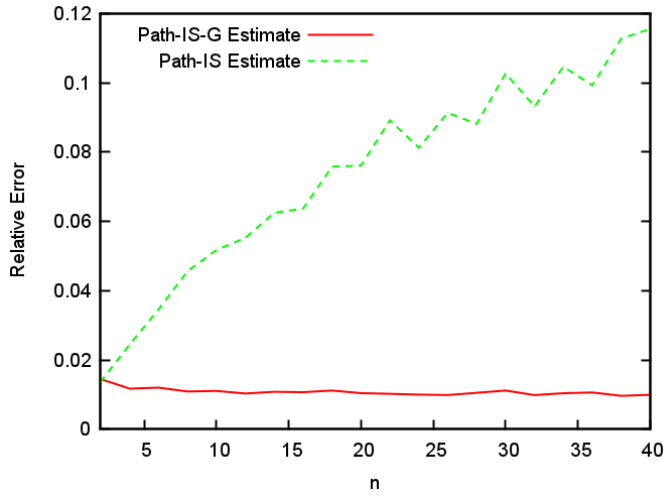


Figure 4.8: Relative errors $\hat{\sigma}/(\hat{\pi}\sqrt{N})$ for the model with one component type. $\lambda = 1/6000$, $\mu = 1$, $\bar{\tau} = 8\,400$ $r_1 = n/2$. We drew $N = 1000$ samples for each data point — the run time for each simulation was less than five seconds.

about 44 seconds to find a solution. In Table 4.10 we combine this result with the results of simulations with a similar run time.

	$\hat{\pi}$ $(\cdot 10^{-4})$	# samples
MC	9.296 ± 0.503	1 410 356
BFB	8.691 ± 0.833	382 539
Path-IS	9.483 ± 0.183	24 319
Path-IS-G	$\mathbf{9.313 \pm 0.105}$	**19 197**
	π $(\cdot 10^{-4})$	# states
PRISM	9.366	512 000

Table 4.10: Unreliability ($\hat{\pi}$) results for the model with 7 component types. $r_i = 2$; $n = 3$, $\lambda = 1/6000$, $\mu = 1$, $\bar{\tau} = 840$.

As expected, Table 4.10 shows that a change of measure based on (4.3) outperforms one based on (4.2). The difference is not dramatic, however. This is because the longer busy cycle durations and higher probability of success during a busy cycle cancel to some extent, causing the rates in the exponential approximation in the two methods to differ by only a factor of 2.2.

Note that the results produced by Path-IS are still usable; however, once we raise $\bar{\tau}$, its performance will again deteriorate. The performance of the two methods does not worsen for higher μ, which can be seen in Table 4.11, in which we raised the value of μ from 1 to 1000. PRISM is still able to produce a result in this situation but only after 10 hours, whereas the run time of the simulation-based methods was 44 seconds. Note also that BFB does not perform very well – the reason is that this model contains so-called *high-probability cycles*, a known complication for BFB. We will discuss high-probability cycles in greater detail in Chapter 5.

In Figure 4.9, the relative error $\hat{\sigma}/(\hat{\pi}\sqrt{N})$ is displayed for increasing values of μ. We can see that the relative error of Path-IS is not only higher but also much more volatile, which shows that the estimate is less reliable. More importantly, the relative error of Path-IS-G remains almost constant; this implies that, unlike Monte Carlo simulation, the time complexity of Path-IS-G does not depend on μ and thereby not on p, the probability that we seek to estimate.

	$\hat{\pi}$ $(\cdot 10^{-7})$	# samples
MC	6.696 ± 13.12	1 413 429
BFB	—	301 322
Path-IS	9.458 ± 0.178	23 872
Path-IS-G	$\mathbf{9.350 \pm 0.103}$	**19 566**
	π $(\cdot 10^{-7})$	# states
PRISM	9.388	512 000

Table 4.11: Unreliability ($\hat{\pi}$) results for the model with 7 component types. $\mu = 1000$; $n = 3$, $r_i = 2$, $\lambda = 1/6000$, $\bar{\tau} = 840$.

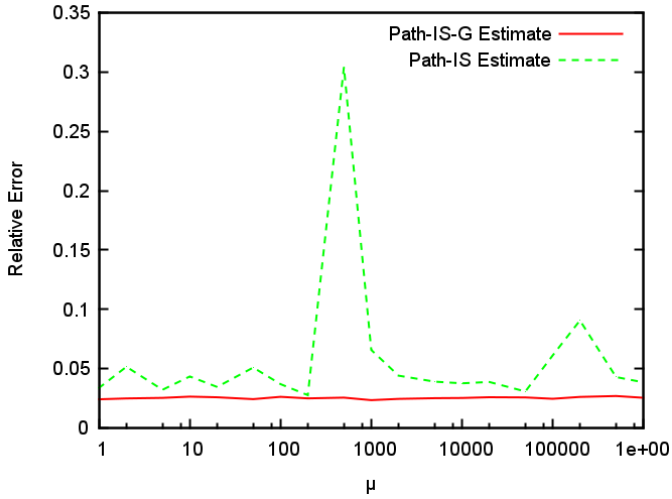


Figure 4.9: Relative errors $\hat{\sigma}/(\hat{\pi}\sqrt{N})$ for the model with 7 component types. $\lambda = 1/6000$, $n = 3$, $\bar{\tau} = 840$, $r_1 = \ldots = r_7 = 2$. $N = 1000$ samples were drawn for each simulation. Note the the peaks are not structural, but simply evidence of the whimsicality of the estimator.

4.3 Shared Repair Facilities

The defining characteristic of the method presented previously is that it is *path-based*; it works well if failures mostly occur in a manner that corresponds to one of the dominant paths. In Section 4.1, we outlined a model class for which we can *guarantee* that this assumption is valid. However, the class of models for which our method (or variations thereof) works well is much broader and contains other models with large practical relevance. To be more specific: we formulated two requirements in Section 4.1 that our system model's repair strategy had to meet: (i) for each component type, repairs had to be handled by a dedicated repair facility, independent of all other component types and (ii) repairs had to start immediately after the first component of a type had failed. In Section 4.2 we have shown how to drop the latter assumption. Additionally, however, one could also think of good reasons to drop the former.

First of all, it is simply unrealistic to assume that two disk sets have two separate repair facilities. A *single* repair facility taking care of all the repairs in the system is often closer to practice. Secondly, models with dependent component types are more theoretically appealing. If all component types are independent, we can compute several important probabilities in the system by first computing them for each component in isolation and then performing a trivial aggregation step.

Of course, generalising to a single repair facility for all component types means

we have to specify a repair *policy* that determines how the repair facility decides which component type to work on whenever components of multiple types are down. From the many policies that have been studied in the literature (see also Chapter 6 of [49]), we will zoom in on one policy, namely true FCFS, to illustrate the wider applicability of our method. True FCFS means that individual components are repaired in the order in which they fail.

The impact of this choice of repair strategy on the state space of our model is such that each state in which several components have failed must be duplicated such that all possible orders of failure can be represented, not discriminating between components of the same type. This has a strong negative effect on the performance of PRISM; not only is expressing this model in the PRISM modelling language a challenge, there is also a dramatic blow-up in the size of the state space. In the benchmark setting, the size of the state space goes from $421\,875$ states to $2\,123\,047\,371$ states.

In Table 4.12, we have displayed simulation results for both the old situation and the new one. Each simulation was run for 33.609 seconds, which is the amount of time needed by PRISM to compute the exact probability in the old setting. The difference between the results of the two repair strategies is marginal: only the path-based IS estimates differ significantly. The reason is that for the system to reach a state in which the two strategies result in different outgoing transition structures, a failure transition must win the race from a repair transition. Since the probability of this happening is assumed to be small for Path-IS-G to work well, it will be hard to notice a difference.

	dedicated repair		1 facility, FCFS	
	$\hat{v}$ $(\cdot 10^{-3})$	# samples	$\hat{v}$ $(\cdot 10^{-3})$	# samples
MC	2.941 ± 0.106	1 009 973	2.956 ± 0.138	596 136
BFB	2.975 ± 0.176	317 989	2.790 ± 0.222	189 032
Path-IS-G	$\mathbf{2.901 \pm 0.020}$	**21 508**	$\mathbf{2.960 \pm 0.027}$	**11 311**
	v $(\cdot 10^{-3})$	# states	v	# states
PRISM	2.928	421 875	N.A.	2 123 047 371

Table 4.12: Unavailability ($\hat{v}$) results when $\lambda = 1/6000$, $\mu = 1$, $n = 2$.

One can check that if we were to switch to deferred repair we would also see that the path-based methods work well if the assumptions are such that there is little difference between dedicated repair and a single repairman. However, if we assume that the components of different types are physically close to each other, this strategy is not only theoretically unappealing but also unrealistic; if there is a single repairman for all component types who would come over to fix broken processors, he would most likely also directly repair a failed disk even if the repair threshold for disks of its type had not been reached.

So consider the following strategy: we again have deferred repair, but when for *any one* component type i a number r_i components have failed, the repairman will come over and begin repair on *all* broken components and leave when the system is

back in the 'all up' state. The repairs are handled in a first-come first-serve fashion irrespective of which component type triggered the repairman to come over.

	$\mu = 1$		$\mu = 1000$	
	$\hat{v}$ (10^{-3})	# samples	$\hat{v}$ (10^{-6})	# samples
MC	1.748 ± 0.061	1 784 962	2.273 ± 2.228	1 759 644
BFB	1.749 ± 0.098	602 621	—	479 383
Path-IS	$\mathbf{1.779 \pm 0.029}$	**20 524**	$\mathbf{1.800 \pm 0.025}$	**20 773**

Table 4.13: Unavailability ($\hat{v}$) results with one repairman who repairs all broken components of all types before leaving in a first-come serve fashion; $\lambda = 1/6000, n = 3, r_i = 2$.

In Table 4.13, we display the simulation results for this setting with $n = 3$, λ and μ equal to the benchmark values and $r_1 = \ldots = r_9 = 2$. We would run into problems if we tried to directly apply the method of Section 4.2 to this situation as even the *typical* set suffers from the state space explosion problem. For example, due to the FCFS policy there are 9! = 362 990 states in which one component of each type has failed, and these are all in the typical set. To circumvent this problem we (incorrectly) assume that for all such states (that differ only in the queue order), the sum of the straight paths leading out of those states will be the same. While this will introduce an error, this error does not get worse as λ gets lower or μ gets smaller, and the resulting estimator remains much better than the standard Monte Carlo estimator whose performance will degrade sharply. In the end, our approximation does not need to be exact for the estimate to remain unbiased. As can be seen in Table 4.13, our method still performs well.

4.4 Conclusions and Discussion

In this section, we first draw the general conclusions. Then we discuss our method's computational complexity and compare it with other methods. Finally, we discuss generalisations of the method and future work.

4.4.1 General Conclusions

In this chapter we have introduced an efficient simulation technique that is able to estimate dependability measures in situations were system failure is a rare event due to high repair rates or low component failure rates. The approach that we used is based on the zero-variance measure for transient failure probabilities in CTMCs, approximated using insight in the most likely paths to failure. We have shown how to apply this method when a more complicated repair strategy such as deferred (or, by analogy, group) repair is used.

We have demonstrated that our technique performs well even for large models as long as the component failure rates are much lower than the repair rates. Also, we have shown that our method performs well in comparison to other methods.

The method of Balanced Failure Biasing combined with forcing is only well-suited for the low λ situations and performs poorly for high μ and n. Numerical techniques, as, e.g., implemented in PRISM, suffer from large state spaces and high uniformisation rates. Table 4.14 summarises this comparison by presenting the best choice of method for given parameter settings.

Performance measure	low λ	high μ	high n
Unreliability	PRISM	Path−IS	Path−IS
Unavailability	PRISM	PRISM	Path−IS

Table 4.14: Which method performs best depends on parametric regime and performance measure.

Note that the method is not specifically designed for high n. If n is too high, Path-IS will also start to see worsening performance.[20] However, for n not too large and λ/μ sufficiently small, Path-IS will have good performance in situations where PRISM runs out of memory. Adapting the simulation for high n situations is part of ongoing research.

4.4.2 Complexity

In Table 4.15, the time and space complexities of the methods whose efficiencies are compared in this chapter are displayed. For the numerical and standard statistical methods the values of the complexities were given in [114] — we rewrote these complexities in terms of the model parameters used in this chapter. The expected time complexity of the statistical methods is inversely proportional to p, the probability that we want to estimate. Using importance sampling, we lose this dependence on p. However, d paths of maximum length n need to be evaluated in each step of the simulation for $O(d)$ potential successor states. Still, Path-IS avoids the two main causes of quickly increasing run time (i.e., the state space explosion problem for the numerical methods and event rarity for standard simulation).

Method	Time Complexity	Space Complexity
Numerical	$O((\lambda+\mu)\bar{\tau}dn^d)$	$O(dn^d)$
Standard Simulation	$O(\lambda\bar{\tau}dp^{-1})$	$O(d)$
Path-IS	$O(\lambda\bar{\tau}d^2n)$	$O(d)$

Table 4.15: Time and space complexities for different methods (repairs always enabled).

In the setting of Section 4.2, the number of paths that needs to be evaluated at each step increases; there are $\prod r_i$ states in the typical set and there are at most $d \cdot \prod r_i$

[20] When the level of redundancy is very high, the rarity of the event may not primarily be caused by the ratio λ/μ but by the fact that there is not enough time for all of the components to fail during the mission time. Fortunately, this is not a common scenario in the analysis of multi-component systems, but it might be encountered in other contexts.

paths inside the typical set that lead to states on the '*edge*' of the typical set. These extra paths lead to an increase in the time complexity of the method. This can be seen by comparing Table 4.15 with Table 4.16.

To speed up Path-IS-G, we used a caching approach: for each state in the typical set we store the sum of the probabilities of the paths to system failure. This results in a higher memory complexity but cuts the time complexity in the typical set back to only $O(\lambda\bar{\tau}dn)$ assuming that lookups in the list of stored probabilities do not contribute to the time complexity (this is possible using cleverly linked lists where each entry points to all states that can be reached in one transition). Outside the typical set, the time complexity is again $O(\lambda\bar{\tau}d^2n)$.

Method	Time Complexity	Space Complexity
Path-IS-G (no caching)	$O(\lambda\bar{\tau}d^3n\prod r_i^2)$	$O(d)$
Path-IS-G (caching)	$O(\lambda\bar{\tau}d^2n)$	$O(d\prod r_i)$

Table 4.16: Time and space complexities for Path-IS-G (repair of type i enabled after r_i components of that type have failed).

4.4.3 Generalisations and Future Work

In this chapter, we have only considered system failures caused by the number of failures of *some* component type i reaching a critical level n_i. The method is also applicable with more general failure conditions, such as simultaneous failure of at least n_i components of each type i. More paths to failure may then need to be considered for a good approximation w — perhaps even a number of paths that increases exponentially in the number of component types — but many of them typically have equal probability which makes accounting for them easier. Similarly, it is straightforward to generalise the extension of Section 4.2 to group repair strategies (see [60]). Another interesting, but more challenging, generalisation is to allow component interdependence, which occurs, for example, when components share repair facilities that follow some predefined repair strategy (see also Section 4.3). Finding the dominant paths in these settings will be the focus of Chapters 5 and 7.

Furthermore, an interesting extension of the method would be to allow the failure and repair time distributions to be any phase-type distribution. Part of the work for this is already done, since the Path-IS-G method discussed in Section 4.2 already considers phase-type distributions in the 'typical' part of the state-space. For the rest of the state-space, a change of measure for the phase-type failure and repair times would be needed; the so-called exponential change of measure [50] can be expected to work well.

Future work could be to extend the method to the case of very large n (see Section 3.2.3), which is an interesting scenario in which numerical methods suffer from the state space explosion problem. Finally, one could add rewards to the model, broadening the applicability of the method to a wide range of real-world cases.

CHAPTER 5

Dominant Paths in General Markov Chains

In Chapter 3, we studied importance sampling techniques for several interesting probabilities in the restricted model class of the birth-death process. One of these probabilities of interest was $\pi_\psi(x_0)$, the probability of entering a rare goal set before a more typical taboo set, starting from x_0. The importance sampling change of measure that we used was based on the dominant paths to the goal set. To obtain an implementable simulation scheme we used the fact that in birth-death processes these paths have a very obvious structure. Particularly, we used the fact that in a birth-death process, one can clearly distinguish between the *births* that lead towards the goal set and the *deaths* that lead away from it, i.e., towards the taboo set. In this chapter we will consider the very general model class of the Discrete-Time Markov Chain (DTMC), in which the structure of these paths is not so obvious. The basic fact that we continue to use is that in order to speed up the simulation procedure, we want to increase the likelihood of moving towards the goal set with respect to moving away from it. In order to do this, we need a notion of distance between states that has meaning in an asymptotic setting where the probability of interest goes to zero.

The fundamental modelling choice made for this chapter is that all rates p_{xz} in the transition probability matrix are parameterised by the so-called *rarity parameter* ϵ. We are interested in the regime $\epsilon \downarrow 0$, in which transitions that depend strongly on ϵ will be taken very rarely — this regime is comparable to the regimes $\lambda \downarrow 0$ and $\mu \to \infty$ for estimating $\pi_\psi(x_0)$ in the birth-death process of Chapter 3. A clear notion of the distance of a path is then its total dependence of the transitions in said path on ϵ, as we will formalise in Section 5.1. Consequently, the dominant paths that we use to construct our change of measure are the paths with the lowest ϵ-distance. We do not require any structure in the dependence of the transitions in the DTMC on ϵ apart from this dependence being polynomial, which means that a priori it is unclear which paths are dominant. In this setting, the method of Balanced Failure Biasing (BFB) was proposed and shown to satisfy Bounded Relative Error (BRE, see Section 5.3) in [108], under the condition that the DTMC in question does not contain so-called high-probability cycles (more on that later). In [80] a combination of BFB and forcing (see Section 1.3.4) was proven to yield an estimator with BRE for π_{ψ^τ}, the probability that we not only reach the goal set before the taboo set but also before a fixed amount of time has passed, in the setting of CTMCs. In this chapter we introduce an algorithm that produces an estimator satisfying the stronger

property of Vanishing Relative Error (VRE, see Section 5.3) for ψ, even when the model contains high-probability cycles, and which, when combined with forcing, also gives an estimator with BRE for $\psi^{\bar{\tau}}$ (in practice it often outperforms BFB).

Part of our approach is to carry out a procedure similar to Dijkstra's algorithm that is run once, before the actual simulation procedure is started, and which determines for the relevant part of the state space the function d, a measure of 'distance' to the goal set, and the function $w^{\diamond}$, an approximation of a state's probability of hitting the goal set before the taboo set. One may argue that if one resorts to using methods on the level of the state space anyway, one may as well use the numerical algorithms of Section 1.1.4 as the runtimes of both Dijkstra's algorithm and the iterative numerical methods such as Gauss-Seidel depend (very roughly) on the number of non-zero entries in the DTMC's transition probability matrix P. However, first of all we will argue in this chapter that our Dijkstra-algorithm does not need to consider the whole state space but only the subset of states that are at most as far away from x_0 in terms of ϵ-distance as the goal set, and their neighbours. This may speed up the algorithm significantly, and may yield an implementable algorithm for DTMCs with an infinite state space. Secondly, some of the results in this section can also be used for algorithms that only use information obtained from a high-level description of the model, as we will see in Chapter 7.

Once the algorithm has determined d and $w^{\diamond}$, we can translate these functions into a well-performing importance sampling change of measure using Zero Variance Approximation (ZVA). As discussed in Section 1.3.5, ZVA boils down to substituting a good approximation w for the true probability into (1.12). In the setting of networks of parallel birth-death processes of Section 4.1.3 we used the approximation w^*, based on the '*straight*' paths. In this section we discuss two candidates for w in the setting of general Markov chains. The first is the most obvious candidate, namely $w^{\diamond}$. The second is ϵ^d, which is more crude than $w^{\diamond}$ but which we use when we only have information of d — this will be the case in Chapter 7.

In addition to determining for each state the distance of the dominant paths to the goal set, our Dijkstra-like procedure also removes all *high-probability cycles*. A high-probability cycle is a path of non-zero length that starts and ends in the same state and which consists only of transitions whose distance in terms of ϵ-orders is zero. While the existence of high-probability cycles can cause BFB to lose BRE, the improved versions of BFB proposed in [60] and [61] are intended to cope with this complication. In [73], which also studies the problem of applying ZVA to reliability models, the models are assumed not to contain high-probability cycles. Because our algorithm runs on the level of the state space anyway, we can remove high-probability cycles without much added effort.

The outline of the rest of this chapter is as follows. In Section 5.1, we will formally describe the modelling assumptions, and introduce the functions d and $w^{\diamond}$ and the two simulation measures. In Section 5.2, we present an algorithm for finding these vectors, based on Dijkstra's algorithm, and prove its correctness. In Section 5.3 we prove that both simulation measures introduced in Section 5.1 satisfy properties that can be shown to imply an estimator with BRE. In Section 5.4, we

discuss how the measure based on w implies the even more desirable property of VRE. In Section 5.5, we obtain an estimator with bounded relative error for π_{ψ^τ}, the probability that we not only reach the goal set before the taboo set but also before a fixed amount of time has passed, in CTMCs. Section 5.6 concludes the chapter. We do not give simulation results in this chapter as the results section of Chapter 6 also includes results obtained using this chapter's algorithm.

5.1 Model Setting

Our model is given in terms of a DTMC with a (large, possibly infinite) state space $\mathcal{X}$. We assume that the system starts in some state $x_0 \in \mathcal{X}$, and that there is (potentially after merging all states in the goal set into a single state) a single rare state $x_b \in \mathcal{X}$. We also assume that there is (again after a potential merge) a single taboo state $x_a \in \mathcal{X}$.[21] We are no longer interested in the behaviour of the system once the system hits x_b or x_a so we assume that these states are absorbing, i.e., have a self-loop with probability 1. Note that the location of x_a and x_b in the state space may *not* be clear a priori; since large DTMCs are typically described using a high-level language such as an SPN (see Section 1.1.3), the shape of the taboo and goal sets may be unclear until the state space is generated explicitly, which we do on-the-fly while running the algorithm described in Section 5.2.

The complete transition probability structure in the DTMC is given by the matrix $(p_{xz})_{x,z\in\mathcal{X}}$, which contains for each pair of states $x, z \in \mathcal{X}$ the probability p_{xz} of jumping from state x to state z — again, in practice this matrix may be given implicitly through a high-level language. The probabilities p_{xz} depend on ϵ, the rarity parameter inherent to the model. Writing $f(x) = \Theta(g(x))$ iff

$$0 < \lim_{x\downarrow 0} \frac{f(x)}{g(x)} < \infty,$$

we assume that $\forall x, z \in \mathcal{X}$,

$$p_{xz}(\epsilon) = \Theta(\epsilon^{r_{xz}}),$$

and we assume, without loss of generality, that $r_{xz} \in \mathbb{N}$.[22] Note that r_{xz} for $x, z \in \mathcal{X}$ are fixed parameters of the model. Examples of DTMCs parameterised in this way can be found in Figures 6.1 and 6.2.

In the remainder of this chapter, we write $f(x) = O(g(x))$ iff

$$\lim_{x\downarrow 0} \frac{f(x)}{g(x)} < \infty,$$

and $f(x) = o(g(x))$ iff

$$\lim_{x\downarrow 0} \frac{f(x)}{g(x)} = 0.$$

[21] In practice this is often a state to which all transitions normally going back to x_0 are routed.

[22] We do need to impose that the set of values of r_{xz} that appear in the model is countable for the proof of Lemma 5.6 to work.

As described in Section 1.1.2, a *path* ω is a sequence $x_0^\omega, x_1^\omega, \ldots, x_{|\omega|}^\omega$ of states in $\mathcal{X}$, with $|\omega|$ equal to the number of steps in the path. We define $\forall x \in \mathcal{X}$

$$\Omega_\psi(x) \triangleq \{\omega : \ x_0^\omega = x, \ \omega \vDash \psi, \ \nexists k < |\omega| \text{ s.t. } (x_0^\omega, x_1^\omega, \ldots, x_k^\omega) \vDash \psi\}, \text{ and}$$

$$\pi_\psi(x) \triangleq \sum_{\omega \in \Omega_\psi(x)} \mathbb{P}(\omega), \tag{5.1}$$

where $\psi = \neg a \, \mathsf{U} \, b$ as defined in Chapter 1. In words, $\Omega_\psi(x)$ is the set of dominant paths from x, and $\pi_\psi(x)$ the probability of the rare event occurring starting in x. We are interested in finding $\pi_\psi(x_0)$. We will do this using *importance sampling*, as described in Section 1.3. That is, we aim to find a simulation measure $\mathbb{Q}$ such that the resulting estimator given by (1.9) has a variance that is as small as possible. To judge whether $\mathbb{Q}$ has good performance in terms of estimator variance, we will investigate whether it satisfies the properties of Bounded Relative Error (BRE) and Vanishing Relative Error (VRE), as defined in Section 1.3.3.

The change of measure technique that we use is ZVA as discussed in Section 1.3.5, meaning that we use (1.13) but use an approximation w instead of π_ψ. Our method for finding a suitable w is to select only a subset of the paths used in the summation of (5.1), namely the dominant paths. In order to determine which paths to select, we will define two related measures for the distance between each state and the rare state x_b. When we consider these distance measures, we assume that the high-probability cycles (as mentioned in the introduction) have already been removed, giving rise to the probability measure $\mathbb{P}'$ (this is discussed in more detail in Section 5.2) First, we define the function $d : \mathcal{X} \to \mathbb{N}$ — intuitively, the smallest number of ϵ's needed to reach x_b— of a state x with respect to ψ as

$$d(x) = \min\{r : \exists \omega \in \Omega_\psi(x) \text{ s.t. } \mathbb{P}'(\omega) = \Theta(\epsilon^r)\}. \tag{5.2}$$

We say that $d(x) = \infty$ if $\Omega_\psi(x)$ is empty. We assume that $d(x_0) > 0$. Furthermore, we also use the function d', defined as

$$d'(x) = \min\{r : \exists \omega \text{ s.t. } x_0^\omega = x_0, x_{|\omega|}^\omega = x, \mathbb{P}'(\omega) = \Theta(\epsilon^r)\}. \tag{5.3}$$

In words, whereas $d(x)$ is the shortest distance between x and x_b, $d'(x)$ is the shortest distance between x and x_0. Next, we define

$$\Omega_\psi^r(x) = \{\omega : \omega \in \Omega_\psi(x), \mathbb{P}'(\omega) = \Theta(\epsilon^r)\}$$

to be the set containing each path ω with $\mathbb{P}(\omega)$ of order r that hits the goal state x_b before the taboo state x_a, and define the function $w^\diamond : \mathcal{X} \to \mathbb{R}^+$ as the probability of the '*dominant*' paths in Ω_ψ:

$$w^\diamond(x) = \sum_{\omega \, \in \, \Omega_\psi^{d(x)}(x)} \mathbb{P}'(\omega).$$

This is the function $w^\diamond$ that we substitute into (1.13) to obtain our change of measure. In general Markov chains, $w^\diamond$ is the analogue of w^* (which is based on the '*straight paths*') for the networks of parallel birth-death processes of Section 4.1.3.

As mentioned in the introduction, we do not need to run the algorithm on the entire state space, we only need the states that are at most as hard to reach from x_0 as x_b and their neighbours (after removing the high-probability cycles). To formalise this, we introduce the following two sets:

$$\Lambda = \{x \in \mathcal{X} : \exists \omega \text{ s.t. } x_0^\omega = x_0,\ x_{|\omega|}^\omega = x,\ \mathbb{P}'(\omega) = O(\epsilon^{d(x_0)})\} \quad \text{and} \tag{5.4}$$

$$\Gamma = \{x \in \mathcal{X} \backslash \Lambda : \exists z \in \Lambda \text{ s.t. } p_{zx} > 0\}.$$

In words: Λ is the relevant part of $\mathcal{X}$, and Γ is the edge '*around*' Λ, i.e., those states to which one can jump directly from Λ. We assume that both sets are finite. The algorithm of this chapter calculates $d(x)$ and $w^\diamond(x)$ only for states $x \in \Lambda \cup \Gamma$. This means that the standard approximation of (1.13) (i.e., using $w^\diamond$ instead of π_ψ) cannot be applied when we are in Γ (after all, not all terms in the sum in the denominator in (1.13) need to be defined in this case). Hence, once we leave Λ we *stop using importance sampling* and revert back to Monte Carlo. A consequence is that the simulation measure $\mathbb{Q}$ is now *non-Markovian*: it is only Markovian until we reach a state outside Λ. Let n be the time at which we hit a state outside Λ. Then $\mathbb{Q}$ is as follows:

$$\mathbb{Q}(x_i^\omega \to x_{i+1}^\omega) = \begin{cases} \dfrac{p_{x_i^\omega x_{i+1}^\omega} w(x_{i+1}^\omega)}{\sum_{z \in \mathcal{X}} p_{x_i^\omega z} w(z)} & \text{if } i < n \\ p_{x_i^\omega x_{i+1}^\omega} & \text{if } i \geq n \end{cases} \tag{5.5}$$

As mentioned in the introduction of this chapter, we have two candidates for w, namely $w^\diamond$ and ϵ^d. While $w^\diamond$ is the best choice, we use ϵ^d in Section 7 because the algorithm presented there only determines d. In fact, in Chapter 7 we determine d for the *entire* state space, meaning that we do not need to distinguish between elements inside and outside Λ. This lack of a distinction does not affect the BRE-property of Chapter 5.3, as will be clear from the proof of Lemma 5.9 (we simply always have that $n = \infty$). As a final note, we assume that for each state x in $\Lambda \cup \Gamma$ it holds that

$$\exists \omega \in \Omega_\psi^{d(x)}(x) \text{ s.t. } x_i^\omega \in \Lambda \cup \Gamma\ \forall i = 1, \ldots, |\omega|. \tag{5.6}$$

Informally, this assumption means that for each state x in Λ or Γ there exists a path of distance $d(x)$ that goes entirely through Λ and Γ. This assumption is necessary for the functions d and $w^\diamond$ that are determined in Algorithm 5.3 to be correct.[23]

In the next section, we discuss the algorithm used to determine d and $w^\diamond$.

[23] A possible way to get around this assumption is to use the approach of Lemma 5.6 where we construct an alternate probability measure $\mathbb{P}''$ in which we reroute all paths that leave the set of interesting states directly to the goal state, and determine d under $\mathbb{P}''$. This is a subject of future research.

5.2 Dijkstra-based Algorithm

In this section, we will describe the algorithm for determining d and w. The algorithm can be broken down into three main routines, specified formally in Algorithms 5.1, 5.2 and 5.3. Unlike Dijkstra's algorithm, the algorithm of this section consists of two phases: a forward phase and a backward phase. The distinction between the phases will be made clear later in this section. The forward phase is described in Algorithm 5.1 and the backward phase is described in Algorithm 5.3. Algorithm 5.2 is a subroutine called by Algorithm 5.1.

The remainder of this section is as follows. We discuss the forward phase in Section 5.2.1 and the backward phase in Section 5.2.2, both by translating the formal description of the algorithm into a more intuitive one. In Section 5.2.3 we prove the correctness of our algorithm.

5.2.1 Forward Phase

In the first phase, we use a procedure based on Dijkstra's well-known algorithm for finding shortest paths in a graph [34] in order to determine d, Λ and to remove all high-probability cycles (as discussed in the introduction of this chapter). In order to detect the high-probability cycles, we use the function $d'(x)$ which represents the distance in terms of low-probability transitions from x_0 to x in $\mathbb{P}$. Note that this is different from d, which is the distance in term of low-probability transitions from x to x_b in $\mathbb{P}'$. We compute d in Algorithm 5.3.

Algorithm 5.1 Forward phase.

Require: Markov chain $(\mathcal{X}, P)$, source x_0, destination x_b.

$\Lambda := \{x_0\}$
$\forall z \in \mathcal{X} \backslash \{x_0\} : d'(z) := \infty$
$d'(x_0) := 0$
$x := x_0$
$P' := P$ ▷ deep copy
while $d'(x) \leq d'(x_b)$ **do**
    $\Lambda := \Lambda \cup \{x\}$
    for all $z \in \mathcal{X}$ s.t. $p_{xz} > 0$ **do**
        $d'(z) := \min(d'(z), d'(x) + r_{xz})$
        if $z \in \Lambda \wedge d'(z) = d'(x)$ **then**
            $P' :=$ loopDetect$((\mathcal{X}, P'), z)$;
        end if
    end for
    $x := \arg\min\{d'(z), z \in \mathcal{X} \backslash \Lambda\}$ ▷ if several states are possible,
end while ▷ any can be chosen
return Λ, P'

Algorithm 5.2 loopDetect($(\mathcal{X}, P), x'$).

Require: Markov chain $(\mathcal{X}, P)$, state x'.

$P' := P$ ▷ deep copy
$A := \{x'\}, B := \{x'\}$
$S_A := \emptyset, S_B := \emptyset$
while $A \backslash S_A \neq \emptyset \wedge B \backslash S_B \neq \emptyset$ **do**
    $\Delta_A := A \backslash S_A, \Delta_B := B \backslash S_B$
    $S_A := S_A \cup A, \forall x \in \Delta_A : \ \forall z \in \mathcal{X}$ s.t. $r_{xz} = 0 : \ A := A \cup \{z\}$
    $S_B := S_B \cup B, \forall x \in \Delta_B : \ \forall z \in \mathcal{X}$ s.t. $r_{zx} = 0 : \ B := B \cup \{z\}$
end while
while $A \backslash S_A \neq \emptyset$ **do**
    $\Delta_A := A \backslash S_A$
    $S_A := S_A \cup A, \forall x \in \Delta_A : \ \forall z \in B$ s.t. $r_{xz} = 0 : \ A := A \cup \{z\}$
end while
while $B \backslash S_B \neq \emptyset$ **do**
    $\Delta_B := B \backslash S_B$
    $S_B := S_B \cup B, \forall x \in \Delta_B : \ \forall z \in A$ s.t. $r_{zx} = 0 : \ B := B \cup \{z\}$
end while
$L := A \cap B$
$D := \{x \in \mathcal{X} \backslash L : \exists z \in L$ s.t. $p'_{zx} > 0\}$ ▷ states to which can be jumped from L
Solve $\begin{bmatrix} \mu_{xz} & = p_{xz} + \sum_{z' \in L} p_{xz'} \mu_{z'z}, & \forall x \in L, z \in D, \\ 1 & = \sum_{z' \in D} \mu_{xz'}, & \forall x \in L \end{bmatrix}$ for μ_{xz}
$p'_{xz} := \mu_{xz}, \forall x \in L, z \in D$
$p'_{xz} := 0, \forall x \in L, z \in L$
return P'.

While running the procedure, we iteratively update Λ — this allows us to use Λ to keep track of the visited states. We initialise $\Lambda = \{x_0\}$ and $d'(x_0) = 0$. We let x_0 be the current state x. Then, we carry out the following routine until x equals x_b: we add each possible successor state z of x to Λ, and set $d'(z) = \min(d'(z), d'(x) + r_{xz})$ — i.e., we let the new best value for $d'(x)$ be the minimum between the old best value and the new possible value. We then set x equal to the element z of Λ that has not been considered before with the lowest $d'(z)$, and start over. When we have reached x_b, we complete the procedure for all states z with $d'(z) = d'(x_b)$ before we terminate the first phase. The set Λ then meets its definition given in (5.4), and for each state $z \in \Lambda$ we have that $d(z) = d(x_b) - d'(z)$.

If, while running the procedure, we find that a state z has a successor state z' such that $d'(z) = d'(z')$, we trigger a loop-detection procedure. The loop-detection procedure essentially boils down to removing all low-probability transitions from the relevant part of the DTMC and finding the Strongly Connected Component (SCC) that contains the states z and z' that triggered the procedure, using the algorithm as described in [12]. Essentially, we determine A, the set of states that can be reached from z using low-probability transitions, and B, the set of states from

Algorithm 5.3 Backward phase.

Require: Markov chain (Λ, P'), end node x_b.

1: $\Lambda' := \emptyset$
2: $\forall z \in \Lambda : \ w^\diamond(z) := 0, d(z) := \infty$
3: $w^\diamond(x_b) := 1, d(x_b) := 0$
4: $x := x_b$
5: $\Gamma := \{x \in \mathcal{X} \backslash \Lambda : \exists z \in \Lambda \text{ s.t. } p_{zx} > 0\}$
6: **while** $\Lambda' \neq \Lambda \cup \Gamma$ **do**
7: $\quad x := z \in (\Lambda \cup \Gamma)\backslash\Lambda' : \begin{cases} d(z) = \min\{d(x') : x' \in (\Lambda \cup \Gamma)\backslash\Lambda'\}, \text{ and} \\ \nexists x' \in (\Lambda \cup \Gamma)\backslash\Lambda' \text{ s.t. } r_{zx'} = 0 \end{cases}$
8: $\quad$ **for all** $z \in \Lambda \cup \Gamma$ **do** ▷ if several states are possible
9: $\quad\quad$ **if** $r_{zx} + d(x) < d(z)$ **then** $w^\diamond(x) := 0$ ▷ in line 7, any can be chosen.
10: $\quad\quad d(z) := \min(d(z), r_{zx} + d(x))$
11: $\quad\quad$ **if** $d(z) = d(x) + r_{zx}$ **then**
12: $\quad\quad\quad w^\diamond(z) := w^\diamond(z) + p'_{zx} w^\diamond(x)$
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad \Lambda' := \Lambda' \cup \{x\}$
16: **end while**
17: **return** $d, w^\diamond, \Gamma$.

which z can be reached using low-probability transitions. The relevant SCC is then $A \cap B$. Note that B is potentially (much) larger than Λ and Γ; it may even be infinite. In order to avoid the algorithm's non-termination due to this complication we alternate between carrying out a step for A and a step for B. If we can no longer find new candidates for A, then A has been determined. Since candidates for L need to be both in A and B, we from then on force candidates for B to also be in A. We then terminate if can no longer find candidates for B in A. A similar argument holds for A and B interchanged. This way, we always terminate in a finite amount of time because A is finite: this will be proven in Section 5.2.3.

Having determined the SCC, we come up with a new Markov chain with the same state space and identical probabilities $\pi_\psi(z)$ in the states, but with transition probabilities around the high-probability cycles redistributed so that the new Markov chain only has low-probability cycles. This can be done using SCC-based state space reduction techniques described in [1], implemented in line 19 of Algorithm 5.2. Removing this cycle gives rise to a new probability matrix P', which is again updated when we find additional loops in a similar manner.

5.2.2 Backward Phase

We begin the second phase in x_b (again, due to the fact that x_b is given implicitly through a high-level description, this may not be trivial without the first phase). We then keep a list Λ', and initialise $\Lambda' = \{x_b\}, w^\diamond(x_b) = 1, d(x_b) = 0$ and $\forall x \in \mathcal{X}, x \neq x_b$

we set $w^\diamond(x) = 0$ and $d(x) = \infty$. For each predecessor x of x_b that is in $\Lambda \cup \Gamma$, we add x to Λ' if this had not been done already and if $d(x) = r_{xx_b}$ we update $w^\diamond(x) := w^\diamond(x) + p'_{xx_b}$. We then choose the next state to consider: this is the state x in $(\Lambda \cup \Gamma)\backslash\Lambda'$ (the set of states that have not yet been considered) for which d is the lowest *and* for which there does not exist another state z in $(\Lambda \cup \Gamma)\backslash\Lambda'$ for which $r_{xz} = 0$. The reason is that otherwise, the probability of the paths going from x to z is never added to x and its predecessors.

We continue performing the same procedure until we have determined $w^\diamond(x)$ for all $x \in \Lambda \cup \Gamma$. Note that we are only able to guarantee that a finite value of d and a non-zero value of w has been assigned to each state in Λ because of the assumption (5.6). The reason is that (5.6) guarantees that for each states in $\Lambda \cup \Gamma$ its shortest paths must pass only through other states in $\Lambda \cup \Gamma$.

5.2.3 Properties of the Algorithm

In this section we prove that our algorithm satisfies a number of desirable correctness properties. Lemma 5.1 states that our loop detection and removal procedures do not impact the probability of interest. Lemma 5.2 states that the functions d and w returned by the algorithm indeed match their definitions. Lemma 5.3 states that the algorithm indeed removes all high-probability cycles. Lemma 5.4 states that the algorithm terminates within a finite amount of time.

Lemma 5.1. *Let $\pi'_\psi(x) = \sum_{\omega \in \Omega_\psi} \mathbb{P}'(\omega)$. Given a discrete-time Markov Chain $(\mathcal{X}, P)$, Algorithm 5.1 returns a Markov chain $(\mathcal{X}, P')$ such that $\forall x \in \mathcal{X}$,*

$$\pi'_\psi(x) = \pi_\psi(x) \tag{5.7}$$

Proof. The matrix P' is constructed iteratively by calls to Algorithm 5.2, so it suffices to show that (5.7) holds for each individual call to Algorithm 5.2. First note that $\pi_\psi(x)$ can be defined using the recurrence relation

$$\pi_\psi(x) = \sum_{z \in S} p_{xz} \pi_\psi(z). \tag{5.8}$$

The matrix P' is defined using the following equations from line 19 of Algorithm 5.2:

$$p'_{xz} = p_{xz} + \sum_{z' \in L} p_{xz'} p'_{z'z}, \qquad \forall x \in L, z \in D \tag{5.9}$$

$$1 = \sum_{z' \in D} p'_{xz'}, \qquad \forall x \in L \tag{5.10}$$

Note that $p_{xz} \neq p'_{xz}$ only if x was part of a high-probability cycle, i.e., an element of the set L defined in line 17 of Algorithm 5.2. Hence, for all states $x \in \mathcal{X}\backslash L$, we have

$$\pi'_\psi(x) = \sum_{z \in \mathcal{X}} p'_{xz} \pi'_\psi(x) = \sum_{z \in \mathcal{X}} p_{xz} \pi'_\psi(x). \tag{5.11}$$

For all states $x \in L$, we have

$$\begin{aligned}
\pi'_\psi(x) &= \sum_{z\in\mathcal{X}} p'_{xz}\pi'_\psi(z) = \sum_{z\in\mathcal{X}\setminus L} p'_{xz}\pi'_\psi(z) \\
&= \sum_{z\in\mathcal{X}\setminus L} \left(p_{xz} + \sum_{z'\in L} p_{xz'}p'_{z'z} \right) \pi'_\psi(z) \\
&= \sum_{z\in\mathcal{X}\setminus L} p_{xz}\pi'_\psi(z) + \sum_{z'\in L} p_{xz'} \sum_{z\in\mathcal{X}\setminus L} p'_{z'z}\pi'_\psi(z) \\
&= \sum_{z\in\mathcal{X}\setminus L} p_{xz}\pi'_\psi(z) + \sum_{z'\in L} p_{xz'}\pi'_\psi(z') \\
&= \sum_{z\in\mathcal{X}} p_{xz}\pi'_\psi(z),
\end{aligned}$$

where the second equality holds because we set $p'_{xz} = 0$ in line 21 of Algorithm 5.2 and the third because of (5.9). Since we now have the same system of linear equations for $\pi'_\psi(x)$ as the one for $\pi_\psi(x)$ given by (5.8), $\pi'_\psi(x)$ and $\pi_\psi(x)$ must be equal, proving the lemma.

□

Lemma 5.2. *The vectors d and w returned by the algorithm satisfy the following requirements $\forall x \in \mathcal{X}$:*

$$d(x) = \min\{r : \exists \omega \in \Omega_\psi(x) \text{ s.t. } \mathbb{P}(\omega) = \Theta(\epsilon^r)\} \tag{5.12}$$

$$w^\diamond(x) = \sum_{\omega \,\in\, \Omega_\psi^{d(x)}(x)} \mathbb{P}(\omega). \tag{5.13}$$

Proof. For (5.12) to be false, one of the following statements must hold:

1. there does not exist a path $\omega \in \Omega_\psi(x)$ such that $\mathbb{P}(\omega) = \Theta(\epsilon^{d(x)})$, or
2. there exists a path $\omega \in \Omega_\psi(x)$ such that $\mathbb{P}(\omega) = \Theta(\epsilon^r)$ with $r < d(x)$.

The first statement is clearly false, as such a path can be found by back-tracking the way $d(x)$ was calculated by the algorithm. The second statement is false because the algorithm is essentially an application of Dijkstra's algorithm on P', and Dijkstra's algorithm returns shortest paths.

For (5.13), we first prove the following invariant: after each step of Algorithm 5.3 it holds that for all states $x \in \Lambda$

$$w^\diamond(x) = \sum_{\substack{\omega \,\in\, \Omega_\psi^{d(x)}(x) \\ x_1^\omega \in \Lambda'}} \mathbb{P}(\omega), \tag{5.14}$$

where $w^\diamond(x)$ is the value after the states in Λ' have been considered, not the final outcome. We will prove this invariant using induction.

Initialisation

After the initialisation, $\Lambda' = \emptyset$ and $w(x_b) = 1$. Since we assumed that x_b had a self-loop with probability 1, this means that the invariant holds.

Induction

Assume that after the ith step, we consider state $x \in \Lambda$. Consider $z \in \Lambda$: by the induction hypothesis, before this step it held that

$$w^\diamond(z) = \sum_{\substack{\omega \in \Omega_\psi^{d(z)}(z) \\ x_1^\omega \in \Lambda' \setminus \{x\}}} \mathbb{P}(\omega).$$

Since we want to prove (5.14), we need to show that what is added to $w(z)$ during this step is exactly

$$\sum_{\substack{\omega \in \Omega_\psi^{d(z)}(z) \\ x_1^\omega \in \Lambda'}} \mathbb{P}(\omega) \; - \sum_{\substack{\omega \in \Omega_\psi^{d(z)}(z) \\ x_1^\omega \in \Lambda' \setminus \{x\}}} \mathbb{P}(\omega) = \sum_{\substack{\omega \in \Omega_\psi^{d(z)}(z) \\ x_1^\omega = x}} \mathbb{P}(\omega).$$

We distinguish two cases: line 11 of Algorithm 5.3 can evaluate to true or false. If it evaluates to false, then $d(z) \neq d(x) + r_{zx}$. In this case it is obvious that no path in $\psi^{d(z)}(z)$ can jump directly to x and since $w^\diamond(z)$ is not updated in this case our invariant will still hold. So assume that line 11 of Algorithm 5.3 evaluates to true; in line 12 of Algorithm 5.3, we add $p'_{zx} w^\diamond(x)$ to $w^\diamond(z)$. By the induction hypothesis,

$$p'_{zx} w^\diamond(x) = p'_{zx} \sum_{\substack{\omega \in \Omega_\psi^{d(x)}(x) \\ x_1^\omega \in \Lambda'}} \mathbb{P}(\omega).$$

Hence, it remains to show that

$$\sum_{\substack{\omega \in \Omega_\psi^{d(z)}(z) \\ x_1^\omega = x}} \mathbb{P}(\omega) = \sum_{\substack{\omega \in \Omega_\psi^{d(x)}(x) \\ x_1^\omega \in \Lambda'}} p'_{zx} \mathbb{P}(\omega).$$

This is true if and only if for each path $(z, x, x_2^\omega, \ldots)$ the following two statements are equivalent:

1. $(z, x, x_2^\omega, \ldots) \in \Omega_\psi^{d(z)}(z)$ and
2. $(x, x_2^\omega, \ldots) \in \Omega_\psi^{d(x)}(x)$ and $x_2^\omega \in \Lambda'$.

First of all, $(x, x_2^\omega, \ldots) \in \Omega_\psi^{d(x)}(x)$ implies $x_2^\omega \in \Lambda'$. After all, if $d(x_2^\omega) < d(x)$ then $d(x_2^\omega)$ must have been considered before x because we always choose the state that has not been considered with the smallest distance to x_b in line 7 of Algorithm 5.3. Otherwise, if $d(x_2^\omega) = d(x)$ then x_2^ω must have been considered because otherwise x could not have been considered as there would exist a state z in $\Lambda \setminus \Lambda'$ s.t. $r_{xz} = 0$

(namely $z = x_2^\omega$) which would contradict the condition imposed upon x in line 7 of Algorithm 5.3.

So, by line 11 of Algorithm 5.3 we know that $d(z) = d(x) + r_{zx}$. This means that if 1) is true, that $(x, x_2^\omega, \ldots)$ must have length $d(x)$, and similarly that if 2) is true, that $(z, x, x_2^\omega, \ldots)$ must have length $d(z)$. Hence, either both are false or both are true, proving the invariant.

□

Lemma 5.3. *For the matrix P' that is returned by Algorithm 5.2 it holds that there are no longer any high-probability cycles. That is, for each sequence $x_0, x_1, \ldots, x_n$ of states for which $x_0 = x_n$, $n > 0$, it holds that*

$$\lim_{\epsilon \downarrow 0} \; p'_{x_0 x_1} \cdot p'_{x_1 x_2} \cdot \ldots \cdot p'_{x_{n-1} x_n} = 0.$$

Proof. Assume the converse of the statement of the lemma. This converse statement is equivalent to saying that there exists a $S \subset \mathcal{X}$ such that all states in this set are reachable from all other states in the set by a high-probability path. Clearly then, for all states x' and x'' in S, $d(x') = d(x'')$. So if one state is considered during the procedure of Algorithm 5.1, then all other states will be considered before the procedure terminates. Consider x^*, the last state of S to be considered. All its successors will be checked, and at least one other state x^{**} in S will be a successor, or x^* could not have been part of the cycle. This successor x^{**} will have its d assigned when it was added to Λ, and $d(x^*) = d(x^{**})$, so in line 11 we call Algorithm 5.2 if had not been earlier. When this happens, all states in S are then identified as part of the loop by Algorithm 5.2. All transitions within this set are then given zero probability, meaning that the high-probability cycle is removed. □

Lemma 5.4. *The algorithm terminates in a finite amount of time.*

Proof. We will consider all steps in the algorithms that could take infinitely long and explain why they do not.

The assignment of $d'(x) = \infty$ for all $x \in \mathcal{X}$ in line 2 of Algorithm 5.1 is done implicitly and does not require knowledge of all states in $\mathcal{X}$.

The while-loop in line 6 of Algorithm 5.1 continues at least until we consider x_b. This happens after a finite number of steps because we only consider other states in Λ before x_b and Λ is finite. Then we continue until the next state x to consider has $d'(x) > d'(x_b)$, which means that this state is outside Λ so this also happens after a finite amount of time because Λ is finite. We only consider each state once because considered states are added to Λ and the next state to be considered must be from $\mathcal{X} \backslash \Lambda$.

For the for-loop in line 8 we only need to consider those states in $\mathcal{X}$ to which we can jump from the state under consideration. The number of states to which we can jump is finite because they must be in Λ and Γ and both are finite.

For the while-loops of Algorithm 5.2, note that the state that triggered the call to Algorithm 5.2 must have been in Λ because we only consider states in Λ in the main routine. Furthermore, A must be finite because each state in A can be reached from

a state in Λ with ϵ-distance zero, and hence would also be in Λ by the definition of Λ in (5.4). So since A is finite the first while loop will terminate in finite time. If the while loop of line 9 or 13 is run it will also terminate in finite time because it only has a finite number of elements from which to select.

The while-loop and the for-loop in Algorithm 5.3 terminate after a finite amount of time because Λ is finite.

Finally, as long as $\Lambda \neq \Lambda'$ line 7 of Algorithm 5.3 always has an element to choose from. After all, assume the converse: that for each state z that have not yet been added to Λ' there exists another state x' such that the transition from z to x' has ϵ-distance zero. This means that if we choose a state, we can always choose a transition with ϵ-distance zero leading to another state not yet considered. Since there is only a finite number of states left to consider, this means that at some end we return to a state seen before. This means that a high-probability cycle exists, which contradicts Lemma 5.3. □

5.3 Bounded Relative Error

In this section we show that the change of measure returned by the algorithm indeed has bounded relative error. We first prove the technical Lemmas 5.5-5.9 before proving the main theorem of this chapter.

Lemma 5.5. *For both $w(x) = w^\diamond(x)$ and $w(x) = \epsilon^{d(x)}$, $x \in \Lambda$, it holds that*

$$w(x) = \Theta(\epsilon^{d(x)}).$$

Proof. It is obvious that $\epsilon^{d(x)} = \Theta(\epsilon^{d(x)})$. For $w^\diamond(x)$, we have that

$$w^\diamond(x) = \sum_{\omega \in \Omega_\psi^{d(x)}(x)} \mathbb{P}(\omega) = \sum_{\omega \in \Omega_\psi^{d(x)}(x)} \Theta(\epsilon^{d(x)}) = \Theta(\epsilon^{d(x)}). \tag{5.15}$$

For the first equality we use Lemma 5.2, and for the third equality we use Lemma 5.3, which together with $|\Lambda| < \infty$ implies that $|\Omega_\psi^{d(x)}(x)| < \infty$. □

Lemma 5.6.

$$\sum_{\omega \in \Omega_\psi(x)} \mathbb{P}'(\omega) = \Theta(\epsilon^{d(x)}).$$

Proof. We will use an argument similar to the one used for Theorem 1 of [73]. With $p_{ij}(\epsilon) = \Theta(\epsilon^{r_{ij}})$ let $\lambda_{ij} = p_{ij}\epsilon^{-r_{ij}}$, and let $\lambda^* = \max\{1, \lambda_{ij} : i, j \in \Lambda\}$. We define the probability measure $\mathbb{P}''$ such that

$$p''_{ij} = \begin{cases} p'_{ij} & \text{if } i \in \Lambda, \\ 1 & i \notin \Lambda \text{ and } j = x_b, \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

Clearly,

$$\sum_{\omega\in\Omega_\psi(x)} \mathbb{P}'(\vec{x}) \leq \sum_{\omega\in\Omega_\psi(x)} \mathbb{P}''(\vec{x}). \tag{5.17}$$

The only technical remark that we make is that some paths that satisfy ψ that had zero probability under $\mathbb{P}'$ now have positive probability. This is not a problem, because our definition of $\Omega_\psi(x)$ in (5.1) does not require that its members have positive probability under $\mathbb{P}$ or $\mathbb{P}'$.

Then,

$$\begin{aligned}
\sum_{\omega\in\Omega_\psi(x)} \mathbb{P}''(\omega) &= \sum_{\omega\in\Omega_\psi^{d(x)}(x)} \mathbb{P}''(\omega) + \sum_{\omega\in\Omega_\psi^{d(x)+1}(x)} \mathbb{P}''(\omega) + \dots \\
&\leq \epsilon^{d(x)} \left[(\lambda^*)^{|\Lambda\cup\Gamma|!}|\Lambda\cup\Gamma|!\right] + \epsilon^{d(x)+1} \left[(\lambda^*)^{|\Lambda\cup\Gamma|!}|\Lambda\cup\Gamma|!\right]^2 + \dots \\
&= \epsilon^{d(x)} \sum_{k=0}^{\infty} \epsilon^k \left[|\Lambda\cup\Gamma|!\cdot(\lambda^*)^{|\Lambda\cup\Gamma|!}\right]^{k+1} \\
&= \epsilon^{d(x)} \frac{\left[|\Lambda\cup\Gamma|!\cdot(\lambda^*)^{|\Lambda\cup\Gamma|!}\right]}{1-\epsilon\left[|\Lambda\cup\Gamma|!\cdot(\lambda^*)^{|\Lambda\cup\Gamma|!}\right]} = \Theta(\epsilon^{d(x)}).
\end{aligned}$$

The inequality holds because of Lemma 5.3, which implies that there can only be a finite number of paths of order $\Theta(\epsilon^k)$ for all $k \in \mathbb{N}$. Also, $\sum_{\omega\in\Omega_\psi(x)} \mathbb{P}'(\omega) \geq w(x) = \Theta(\epsilon^{d(x)})$ (see Lemma 5.5), meaning that $\sum_{\omega\in\Omega_\psi(x)} \mathbb{P}'(\omega)$ is both upper and lower bounded by something that is $\Theta(\epsilon^{d(x)})$, proving the lemma. □

Lemma 5.7. *If $w(x) = \Theta(\epsilon^{d(x)})$, then for all $x \in \Lambda$ we have that*

$$\sum_{z\in\mathcal{X}} p_{xz} w(z) = \sum_{z\in\Lambda\cup\Gamma} p_{xz}\Theta(\epsilon^{d(z)}) = \Theta(\epsilon^{d(x)}).$$

Proof. Since $\Lambda \cup \Gamma$ has a finite number of elements, the ϵ-order of the sum equals the ϵ-order of its largest element. Let z' be the state such that $p_{xz'}\Theta(\epsilon^{d(z')})$ has the lowest ϵ-order in the sum. Assume that its ϵ-order is smaller than $d(x)$, then there exists a path from x via z' to x_b with cost lower than $d(x)$, which contradicts the definition of $d(x)$ as the ϵ-order of the shortest path from x to x_b. This proves the lemma. □

Lemma 5.8. *Let $n = \min\{i : x_i^\omega \notin \Lambda \vee x_i^\omega = x_b\}$. Let*

$$l_n(\omega) = r_{x_0^\omega x_1^\omega} + \dots + r_{x_{n-1}^\omega x_n^\omega}.$$

Then if $w(x) = \Theta(\epsilon^{d(x)})$, we have for all paths $\Omega \in \Omega_\psi(x_0)$ that

$$\mathbb{P}(\omega)\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = O\left(\epsilon^{l_n(\omega)+d(z)}\right).$$

Proof. First, note that

$$\mathbb{P}(\omega) = O\left(\epsilon^{l_n(\omega)+d(x_n^\omega)}\right).$$

The reason is that the ϵ-distance crossed to reach x_n^ω equals $l_n(\omega)$ and that $d(x_n^\omega)$ is the shortest ϵ-distance needed to reach x_b from x_n^ω. As for the likelihood ratio $\mathbb{P}/\mathbb{Q}$, we have that

$$\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = \prod_{i=1}^{|\omega|} \frac{p_{x_{i-1}^\omega x_i^\omega}}{q_{x_{i-1}^\omega x_i^\omega}} = \left(\prod_{i=1}^{n} \frac{p_{x_{i-1}^\omega x_i^\omega}}{q_{x_{i-1}^\omega x_i^\omega}}\right) \cdot \left(\prod_{i=n+1}^{|\omega|} \frac{p_{x_{i-1}^\omega x_i^\omega}}{q_{x_{i-1}^\omega x_i^\omega}}\right) \tag{5.18}$$

For the first part of (5.18), we have that

$$\begin{aligned}\prod_{i=1}^{n} \frac{p_{x_{i-1}^\omega x_i^\omega}}{q_{x_{i-1}^\omega x_i^\omega}} &= \prod_{i=1}^{n} \frac{\sum_{z\in\mathcal{X}} p_{x_{i-1}^\omega z} w(z)}{w(x_i^\omega)} \\ &= \prod_{i=1}^{n} \frac{\Theta(\epsilon^{d(x_{i-1}^\omega)})}{\Theta(\epsilon^{d(x_i^\omega)})} = \frac{\Theta(\epsilon^{d(x_0)})}{\Theta(\epsilon^{d(x_n^\omega)})}.\end{aligned}$$

The first equality follows directly from (5.5) and the second equality from the lemma's assumption and Lemmas 5.5 and 5.7. For the second part of (5.18) we have that

$$\left(\prod_{i=n+1}^{|\omega|} \frac{p_{x_{i-1}^\omega x_i^\omega}}{q_{x_{i-1}^\omega x_i^\omega}}\right) = 1$$

because we continue with standard Monte Carlo after time n. Combining all this, we have that

$$\mathbb{P}(\omega)\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = O\left(\epsilon^{l_n(\omega)+d(x_n^\omega)}\right) \cdot \Theta\left(\epsilon^{d(x_0)-d(x_n^\omega)}\right) = O\left(\epsilon^{l_n(\omega)+d(x_0)}\right),$$

proving the lemma. □

Lemma 5.9. *If* $w(x) = \Theta(\epsilon^{d(x)})$, *then*

$$\sum_{\omega\in\Omega_\psi(x_0)} \mathbb{P}(\omega)\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = \Theta(\epsilon^{2d(x_0)}) \tag{5.19}$$

Proof. By Lemma 5.8, we know that

$$\sum_{\omega\in\Omega_\psi(x_0)} \mathbb{P}(\omega)\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} = \sum_{\omega\in\Omega_\psi(x_0)} \Theta(\epsilon^{d(x_0)+l_n(\omega)}). \tag{5.20}$$

with n and l_n as defined in the statement of Lemma 5.8. We can then show (5.19) using an argument similar to the one made to prove Lemma 5.6, given informally below. First, by the definition of Λ and d, we know that there exists at least one

path from x_0 to x_b with cost $d(x_0)$ and that this path never leaves Λ — hence, for this path $l_n(\omega) = d(x_0)$. So the sum in (5.20) is lower bounded by something that is $\Theta(\epsilon^{2d(x_0)})$.

To establish the upper bound, note that we can decompose the last sum in (5.20) over all paths in $\Omega_\psi(x_0)$ into a countable number of sums over $\Omega_\psi^{d(x_0)+k}(x_0)$ for $k = 0, 1, \ldots$ because $r_{xz} \in \mathbb{N}$ for all $x, z \in \mathcal{X}$. Each term has a finite number of elements because of the lack of high-probability cycles, allowing us to upper bound the last sum in (5.20) by a geometric series that converges to $\Theta(\epsilon^{2d(x_0)})$.

□

Theorem 5.1. *If $w(x) = \Theta(\epsilon^{d(x)})$, the estimator based on* (5.5) *has the Bounded Relative Error property.*

Proof. As defined in Section 1.3.3, our estimator has BRE if the relative error (defined as the ratio of the standard deviation of the estimator to its mean) is bounded from above by a finite constant. Note that this is equivalent to saying that the square of the relative error, given by

$$\frac{\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)}{\mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)} = \frac{\mathbb{E}_{\mathbb{Q}}(L^2_{\mathbb{Q}} \cdot \mathbf{1}_\psi) - \mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)}{\mathbb{E}^2_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)} = \frac{\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)}{\mathbb{E}^2_{\mathbb{P}}(\mathbf{1}_\psi)} - 1 \tag{5.21}$$

is bounded from above by a finite constant. We have that

$$\frac{\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)}{\mathbb{E}^2_{\mathbb{P}}(\mathbf{1}_\psi)} = \frac{\sum_{\omega \in \psi(x_0)} \mathbb{P}(\omega) \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}}{\left(\sum_{\omega \in \psi(x_0)} \mathbb{P}(\omega) \right)^2} = \frac{\sum_{\omega \in \psi(x_0)} \mathbb{P}(\omega) \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}}{(\Theta(\epsilon^{d(x_0)}))^2} = \frac{\Theta(\epsilon^{2d(x_0)})}{\Theta(\epsilon^{2d(x_0)})} = \Theta(1),$$

as the second equality follows from Lemma 5.6 and the third from Lemma 5.9. The squared relative error equals this quantity minus one, and is hence also bounded above by a finite constant as ϵ goes to zero. Since the probability of interest goes to zero while the relative error stays bounded, we have BRE, proving the theorem. □

By Lemma 5.5, we then know that we obtain an estimator with the BRE-property if we use ϵ^d or $w^\diamond$ for the approximation w in (5.5).

5.4 Vanishing Relative Error

In the previous section, we showed that we can use either $w^\diamond$ or only d (through ϵ^d) to obtain an estimator with the BRE-property. Since $w^\diamond$ is obviously a much more sophisticated approximation for π_ψ, this raises the question of whether we can establish stronger results for $w^\diamond$ than just bounded relative error. The purpose of this section is to do just that: we will prove that an estimator using $w^\diamond$ has the property of *vanishing* relative error.

Formally, we need the following two additional lemmas for VRE.

Lemma 5.10.

$$\sum_{\omega \in \Omega_\psi(x)} \mathbb{P}(\omega) = w^\diamond(x) + o(\epsilon^{d(x)})$$

Proof. We know from Section 5.2.3 that $w^\diamond(x) = \sum_{\omega \in \Omega_\psi^{d(x)}(x)} \mathbb{P}(\omega)$, and we can use an argument involving a geometric sum similar to what is used in the proof of Lemma 5.6 to show that the sum of the probabilities of all other paths is $O(\epsilon^{d(x)+1})$ and hence $o(\epsilon^{d(x)})$, proving the lemma. □

Lemma 5.11.

$$\sum_{z \in \Lambda} p_{xz} w^\diamond(z) = w^\diamond(x) + o(\epsilon^{d(x)})$$

Proof. Note that

$$\begin{aligned} \sum_{z \in \Lambda} p_{xz} w^\diamond(z) &= \sum_{z \in \Lambda} p_{xz} \sum_{\omega \in \Omega_\psi^{d(z)}(z)} \mathbb{P}(\omega) = \sum_{\omega \in \Omega_\psi^{d(x)}(x)} \mathbb{P}(\omega) + o(\epsilon^{d(x)}) \\ &= w^\diamond(x) + o(\epsilon^{d(x)}). \end{aligned}$$

The second equality holds because for a path to be a shortest path from x to x_b, it must jump to a neighbouring state z of x and then take a shortest path from z. This proves the lemma.

□

With these lemmas, we can establish Theorem 5.2.

Theorem 5.2. *The estimator based on* (5.5)*, with $w \equiv w^\diamond$ as returned by the algorithm of Section 5.2, has the Vanishing Relative Error property.*

Proof. The definition for VRE as given in Section 1.3.3 is that as the probability of interest goes to zero, the relative error goes to zero along with it. By (5.21), we know that this means that the quantity $\mathbb{E}_\mathbb{P}(L_\mathbb{Q} \cdot \mathbf{1}_\psi)/\mathbb{E}^2_\mathbb{P}(\mathbf{1}_\psi)$ goes to one as $\epsilon \downarrow 0$. Note that

$$\begin{aligned} \lim_{\epsilon \downarrow 0} \frac{\mathbb{E}_\mathbb{P}(L_\mathbb{Q} \cdot \mathbf{1}_\psi)}{\mathbb{E}^2_\mathbb{P}(\mathbf{1}_\psi)} &= \lim_{\epsilon \downarrow 0} \frac{\epsilon^{-2d(x_0)}}{\epsilon^{-2d(x_0)}} \cdot \frac{\mathbb{E}_\mathbb{P}(L_\mathbb{Q} \cdot \mathbf{1}_\psi)}{\big(w(x_0) + o(\epsilon^{d(x_0)})\big)^2} \\ &= \frac{\lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \cdot \mathbb{E}_\mathbb{P}(L_\mathbb{Q} \cdot \mathbf{1}_\psi)}{\lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \cdot \big(w^2(x_0) + o(\epsilon^{2d(x_0)})\big)}. \end{aligned} \tag{5.22}$$

The first equality follows from Lemma 5.10. To see why the second equality is justified, note that by Lemma 5.5 we have that $w^\diamond(x) = \Theta(\epsilon^{d(x)})$, so that the denominator in the final expression of (5.22) is $\Theta(1)$. Also, we know by Theorem 5.1 that the ratio is $\Theta(1)$, meaning that the numerator must also be $\Theta(1)$. Since both the numerator and denominator are $\Theta(1)$, we can replace the limit of quotients with a quotient of limits, justifying (5.22).

For the denominator of (5.22) we obviously have that

$$\lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \cdot \left(w^2(x_0) + o(\epsilon^{2d(x_0)}) \right) = \epsilon^{-2d(x_0)} \cdot w^2(x_0), \tag{5.23}$$

so we focus on the numerator, given by

$$\lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \cdot \mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi}) = \lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \sum_{\Omega_{\psi}(x_0)} \mathbb{P}(\omega) \cdot \left(\prod_{i=1}^{n} \frac{p_{x_{i-1}^{\omega} x_i^{\omega}}}{q_{x_{i-1}^{\omega} x_i^{\omega}}} \right). \tag{5.24}$$

Note that using Lemma 5.11 and a line of reasoning similar to the one used to establish Lemma 5.9 we have that

$$\left(\prod_{i=1}^{n} \frac{p_{x_{i-1}^{\omega} x_i^{\omega}}}{q_{x_{i-1}^{\omega} x_i^{\omega}}} \right) = \prod_{i=1}^{n} \frac{w(x_{i-1}^{\omega}) + o(\epsilon^{d(x_{i-1}^{\omega})})}{w(x_i^{\omega})}$$

As for

$$\prod_{i=1}^{n} \left(w(x_{i-1}^{\omega}) + o(\epsilon^{d(x_{i-1}^{\omega})}) \right),$$

note that this equals a sum in which each term is a product of n terms in which the ith term of the product is either $\Theta(\epsilon^{d(x_{i-1}^{\omega})})$ or $o(\epsilon^{d(x_{i-1}^{\omega})})$. The dominant term is $\prod_{i=1}^{n} w(x_{i-1}^{\omega})$ and the remainder is $o(\epsilon^{\sum_{i=1}^{n} d(x_{i-1}^{\omega})})$. In summary, we have that

$$\begin{aligned} \left(\prod_{i=1}^{n-1} \frac{p_{x_{i-1}^{\omega} x_i^{\omega}}}{q_{x_{i-1}^{\omega} x_i^{\omega}}} \right) &= \frac{\left(\prod_{i=1}^{n} w(x_{i-1}^{\omega}) \right) + o(\epsilon^{\sum_{i=1}^{n} d(x_{i-1}^{\omega})})}{\prod_{i=1}^{n-1} w(x_i^{\omega})} \\ &= w(x_0) + o(\epsilon^{d(x_0)}). \end{aligned}$$

Combining this with (5.24) and Lemma 5.10 gives us

$$\begin{aligned} \lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \cdot \mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi}) &= \lim_{\epsilon \downarrow 0} \epsilon^{-2d(x_0)} \sum_{\omega \in \Omega_{\psi}(x_0)} \mathbb{P}(\omega) \cdot \left(w(x_0) + o(\epsilon^{d(x_0)}) \right) \\ &= \epsilon^{-2d(x_0)} w^2(x_0), \end{aligned}$$

which together with (5.22) and (5.23) implies that $\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi}) / \mathbb{E}_{\mathbb{P}}^2(\mathbf{1}_{\psi})$ goes to one as $\epsilon \downarrow 0$, which in turn implies VRE. □

5.5 Time-Bounded Probabilities

So far, we have described an approach for estimating the probability of observing a path that satisfies ψ, i.e., reaching x_b before x_a in a DTMC. To obtain an efficient estimator, we use the shortest paths that satisfy ψ. In this section, we investigate whether the same approach can be used to efficiently estimate $\psi^{\bar{\tau}}$, i.e., reaching x_b before x_a *before* $\bar{\tau}$ time units have passed, in a CTMC. In this setting, it still holds

that the event of interest is rare if transitions taking the system closer to the taboo set are much more likely than transitions that lead to the goal set. However, the added complication is that it is now also a rare event to leave states with an exit rate that vanishes as $\epsilon \downarrow 0$ before $\bar{\tau}$ has passed. To make this concrete, note that for the path property ψ in a DTMC we have for all $x \in \mathcal{X}$ that

$$\pi_\psi(x) = \sum_{\omega \in \Omega_\psi(x)} \mathbb{P}(\omega) \cdot \mathbb{P}(\psi|\omega) = \sum_{\omega \in \Omega_\psi(x)} \mathbb{P}(\omega)$$

where the second equality holds because $\mathbb{P}(\psi|\omega) = 1$ for each path $\omega \in \Omega_\psi(x)$. However, in a CTMC we have that $\mathbb{P}(\psi^{\bar{\tau}}|\omega)$ does *not* equal 1 for each path $\omega \in \Omega_\psi(x)$ — instead, it equals the distribution function of a sum of $|\omega|$ exponentially distributed random variables where the mean of the ith random variable equals $1/\eta(x^\omega_{i-1})$. When for all states $x \in \mathcal{X}$ it holds that $\eta(x) = \Theta(1)$, one easily verifies that the BRE-property still holds even if all sojourn times are drawn using standard Monte Carlo (although VRE will no longer hold). However, when it is no longer the case we need to adjust our approach.

From now on, we assume that for all $x \in \mathcal{X}$ we have that

$$\eta(x) = h(x) \cdot \epsilon^{\rho_x},\ \rho_x \in \mathbb{N}$$

such that $h(x)$ is independent of ϵ. It now holds that if for some x in ω we have that $\rho_x > 0$, then $\mathbb{P}(\bar{\tau}|\omega)$ goes to zero as $\epsilon \downarrow 0$. Before we continue our analysis, it is important to know what the effect is of the values ρ_x, $x \in \omega$, is on the speed with which $\mathbb{P}(\bar{\tau}|\omega)$ goes to zero. This is the subject of Lemma 5.12.

Lemma 5.12. *Let the random variables $X_1, \ldots, X_n$ be exponentially distributed such that $\mathbb{E}(X_i) = \frac{1}{\eta_i(\epsilon)}$, with $\eta_i(\epsilon) = h_i \epsilon^{\rho_i}$, $h_i > 0$ and $\rho_i \in \mathbb{N}$ for $i = 1, \ldots, n$. Let $S = X_1 + \ldots + X_n$ have cumulative distribution function $F_S : \mathbb{R}^+ \to [0, 1]$, and let $k = \sum_{i=1}^n \rho_i$. Then it holds $\forall \bar{\tau} \in \mathbb{R}^+$ that*

$$F_S(x) = \Theta\left(\epsilon^k\right).$$

Proof. Note that

$$F_S(x) = \int_0^{\bar{\tau}} \int_0^{\bar{\tau}-y_1} \cdots \int_0^{\bar{\tau}-\sum_1^{n-1} y_i} \left(\prod_{i=1}^n h_i\right) \cdot e^{-\sum_{i=1}^n h_i y_i} dy_n \cdots dy_1,$$

so that

$$\lim_{\epsilon \downarrow 0} \frac{F_S(x)}{\epsilon^k} = \lim_{\epsilon \downarrow 0} \left(\prod_{i=1}^n \frac{h_i}{\epsilon^k}\right) \cdot \lim_{\epsilon \downarrow 0} \int_0^{\bar{\tau}} \cdots \int_0^{\bar{\tau}-\sum_1^{n-1} y_i} e^{-\sum_{i=1}^n h_i y_i} dy_n \cdots dy_1. \quad (5.25)$$

The exchange of limit of products to product of limits in (5.25) is justified because the first term in the second expression is $\Theta(1)$. Hence, it is left to show that the

second term in the second expression of (5.25) is also $\Theta(1)$. Note that because $e^{-\sum_{i=1}^{n} h_i y_i}$ is positive, we have that

$$\int_0^{\bar{\tau}} \int_0^{\bar{\tau}-y_1} \cdots \int_0^{\bar{\tau}-\sum_1^{n-1} y_i} e^{-\sum_{i=1}^{n} h_i y_i} dy_n \cdots dy_1 \leq \int_0^{\bar{\tau}} \int_0^{\bar{\tau}} \cdots \int_0^{\bar{\tau}} e^{-\sum_{i=1}^{n} h_i y_i} dy_n \cdots dy_1 = \Theta(1).$$

Also,

$$\int_0^{\bar{\tau}} \int_0^{\bar{\tau}-y_1} \cdots \int_0^{\bar{\tau}-\sum_1^{n-1} y_i} e^{-\sum_{i=1}^{n} h_i y_i} dy_n \cdots dy_1 \geq \int_0^{\frac{\bar{\tau}}{n}} \int_0^{\frac{\bar{\tau}}{n}} \cdots \int_0^{\frac{\bar{\tau}}{n}} e^{-\sum_{i=1}^{n} h_i y_i} dy_n \cdots dy_1 = \Theta(1)$$

because the region of integration in the second expression is contained in the region of integration in the first expression (see also the proof of Lemma 2 of [80]). Hence, the second term in the second expression of (5.25) is bounded from below and above by something that is $\Theta(1)$, meaning that it must be $\Theta(1)$ itself, proving the lemma. □

Lemma 5.12 tells us that the asymptotic distances r of the transition probabilities can be compared to the asymptotic distances ρ of the exit rates. I.e., whereas we had in the DTMC-setting that the distance of a path ω equalled

$$\sum_{i=0}^{|\omega|-1} r_{x_i^\omega x_{i+1}^\omega},$$

we now have that the distance of a timeless path ω equals

$$\sum_{i=0}^{|\omega|-1} \left(r_{x_i^\omega x_{i+1}^\omega} + \rho_{x_i^\omega} \right).$$

Another way to look at this is to say that the distance of a path is measured by the sum of the ϵ-distances of its CTMC-rates $p_{xz}\eta(x)$. This gives rise to the new distance measure $d^{\bar{\tau}} : \mathcal{X} \to \mathbb{N}$, defined as

$$d^{\bar{\tau}}(x) = \min\{r : \exists \omega \in \Omega_\psi(x) \text{ s.t. } \mathbb{P}(\omega) \cdot \mathbb{P}(\psi^{\bar{\tau}}|\omega) = \Theta(\epsilon^r)\}.$$

We can determine $d^{\bar{\tau}}$ by running a version of Algorithm 5.1 in which each instance of r_{xz} is replaced by $r_{xz} + \rho_x$. We can then establish the following lemma.

Lemma 5.13.

$$\sum_{\omega \in \Omega_\psi(x)} \mathbb{P}'(\omega) \cdot \mathbb{P}(\psi^{\bar{\tau}}|\omega) = \Theta(\epsilon^{d^{\bar{\tau}}(x)}).$$

Proof. Identical to the proof of Lemma 5.6. □

Taking all this into account, we can establish the following theorem.

Theorem 5.3. *If $w(x) = \Theta(\epsilon^{d^{\bar{\tau}}(x)})$, the estimator for $\psi^{\bar{\tau}}$ based on* (5.5) *for the transition probabilities combined with the sojourn times sampled using forcing as described in Section 1.3.4 has the Bounded Relative Error property.*

Proof. Using arguments similar to the ones used in the proof of Theorem 5.1, we have BRE in this setting if

$$\frac{\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi^{\bar{\tau}}})}{\mathbb{E}^2_{\mathbb{P}}(\mathbf{1}_{\psi^{\bar{\tau}}})} \tag{5.26}$$

is bounded from above by something that is $\Theta(1)$. Since we have by Lemma 5.13 that the denominator is $\Theta(\epsilon^{2d^{\bar{\tau}}(x_0)})$, we show that the same holds for the numerator. Note that

$$\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi^{\bar{\tau}}}) = \sum_{\omega \in \Omega_\psi(x_0)} \mathbb{P}(\omega) \cdot \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} \cdot \mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi^{\bar{\tau}}} | \omega).$$

By (1.11), we also have for all $\omega \in \Omega_\psi(x_0)$ that

$$\begin{aligned}
\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_{\psi^{\bar{\tau}}} | \omega) &= \int_0^{\bar{\tau}} \cdots \int_0^{\bar{\tau} - \sum_1^{|\omega|-1} y_i} \prod_{i=1}^{|\omega|} \frac{\eta(x^\omega_{i-1}) e^{-\sum_{i=1}^{|\omega|} \eta(x^\omega_{i-1}) y_i}}{\left(1 - e^{-\eta(x^\omega_{i-1})(\bar{\tau} - \sum_{j=1}^{i-1} y_j)}\right)^{-1}} dy_{|\omega|} \cdots dy_1 \\
&\leq \int_0^{\bar{\tau}} \cdots \int_0^{\bar{\tau} - \sum_1^{|\omega|-1} y_i} \prod_{i=1}^{|\omega|} \frac{\eta(x^\omega_{i-1}) e^{-\sum_{i=1}^{|\omega|} \eta(x^\omega_{i-1}) y_i}}{\left(1 - e^{-\eta(x^\omega_{i-1})\bar{\tau}}\right)^{-1}} dy_{|\omega|} \cdots dy_1 \\
&= \prod_{i=1}^{|\omega|} \left(1 - e^{-\eta(x^\omega_{i-1})\bar{\tau}}\right) \cdot \mathbb{E}(\mathbf{1}_{\psi^{\bar{\tau}}} | \omega)
\end{aligned}$$

Let the sets I and J be such that $I \cup J = \{1, \ldots, |\omega|\}$ and that $i \in I$ iff $\eta(x^\omega_{i-1})$ is independent of ϵ and that $i \in J$ otherwise. Also, let $k(\omega) = \rho_{x^\omega_1} + \ldots + \rho_{x^\omega_{|\omega|}}$. Then

$$\begin{aligned}
\lim_{\epsilon \downarrow 0} \epsilon^{-k(\omega)} \prod_{i=1}^{|\omega|} \left(1 - e^{-\eta(x^\omega_{i-1})\bar{\tau}}\right) &= \prod_{i \in I} \left(1 - e^{-\eta(x^\omega_{i-1})\bar{\tau}}\right) \\
&\quad \cdot \lim_{\epsilon \downarrow 0} \epsilon^{-k(\omega)} \prod_{i \in J} \left(1 - e^{-\eta(x^\omega_{i-1})\bar{\tau}}\right).
\end{aligned} \tag{5.27}$$

Hence, with $h^* = \max\{1, h(x^\omega_{i-1}) : i \in J\}$,

$$\begin{aligned}\prod_{i\in J}\left(1-e^{-\eta(x^\omega_{i-1})\bar\tau}\right) &= \prod_{i\in J}\left(\sum_{j=1}^{\infty}\frac{(-1)^{j-1}}{j!}\cdot\left(\eta(x^\omega_{i-1})\cdot\bar\tau\right)^j\right)\\ &\le \prod_{i\in J}\left(\sum_{j=1}^{\infty}\left(\eta(x^\omega_{i-1})\cdot\bar\tau\right)^j\right)\\ &\le \sum_{m=0}^{\infty}(h^*)^{|J|+m}\cdot\epsilon^{k(\omega)+m}\cdot(|J|)^m\cdot\bar\tau^{|J|+m}\\ &= \frac{\epsilon^{k(\omega)}\cdot(h^*)^{|J|}\cdot\bar\tau^{|J|}}{1-\epsilon\cdot h^*\cdot|J|\cdot\bar\tau} = \Theta(\epsilon^{k(\omega)}),\end{aligned}$$

where the first equality follows form the Maclaurin series expansion of e^y, $y \in \mathbb{R}^+$, the first inequality from the fact that $(-1)^j/j! \le 1$ for $j = 1, 2, \ldots$, the second inequality from a partitioning of the product of sums with regard to the number of ϵ-orders, and the second equality because due to the limit of $\epsilon \downarrow 0$ we can choose ϵ small enough for the series to converge. Together with (5.27), this implies that

$$\prod_{i=1}^{|\omega|}\left(\sum_{j=1}^{\infty}\frac{(-1)^{j-1}}{j!}\left(\eta(x^\omega_{i-1})\bar\tau\right)^j\right) = \Theta(\epsilon^{k(\omega)}).$$

Since we have that $\mathbb{E}(\mathbf{1}_{\psi^{\bar\tau}}|\omega) = \mathbb{P}(\psi^{\bar\tau}|\omega)$, and that $\psi^{\bar\tau}$ given a path ω in $\Omega_\psi(x_0)$ is simply the event that a sum of exponentially distributed random variables does not exceed $\bar\tau$, we can apply Lemma 5.12 to obtain

$$\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi^{\bar\tau}}|\omega) = \Theta(\epsilon^{k(\omega)})\cdot\mathbb{E}(\mathbf{1}_{\psi^{\bar\tau}}|\omega) = O(\epsilon^{2k(\omega)}).$$

Combining all this, we have, with $l(\omega) = r_{x^\omega_0 x^\omega_1} + \ldots + r_{x^\omega_{|\omega|-1}x^\omega_{|\omega|}}$, that

$$\begin{aligned}\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi^{\bar\tau}}) &\le \sum_{\omega\in\Omega_\psi(x_0)}\mathbb{P}(\omega)\cdot\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\cdot\Theta(e^{2k(\omega)})\\ &= \sum_{\omega\in\Omega_\psi(x_0)}\Theta(e^{d(x_0)+l_n(\omega)+2k(\omega)}) = \Theta(\epsilon^{2d^{\bar\tau}(x_0)}),\end{aligned} \tag{5.28}$$

where the second equality follows from Lemma 5.8 (l_n is defined as in the statement of that lemma). The third equality follows from a geometric sum argument similar to the one made for Lemma 5.6 (i.e., by the definition of $d^{\bar\tau}$, there is at least one path ω such that $d(x_0) + d'(x^\omega_n) + 2k(\omega) = 2d^{\bar\tau}(x_0)$, and we can then partition the sum in the third expression in (5.28) into a countable series of converging sums, allowing us to upper bound the entire sum by something that is $\Theta(\epsilon^{2d^{\bar\tau}(x_0)})$). In doing so, it proves the theorem.

□

5.6 Conclusion

In this chapter, we have presented an algorithm for obtaining a well-performing simulation measure for two interesting probabilities, namely the probability of hitting a rare goal set before a typical set in general Markov chains and the time-bounded version of this probability for CTMCs. The most restrictive assumption made on the models is the one of (5.6). However, we hope to resolve the need for this assumption in further research. Numerical results for the algorithm in this chapter are presented in the Chapter 6.

CHAPTER 6

Variance Reduction for Free

In this chapter, we present two techniques that use knowledge about which paths are dominant in order to potentially achieve further variance reduction than just the gain resulting from applying IS. The idea is that this knowledge is obtained '*for free*' when we run the algorithm used to construct the change of measure of Chapter 5 (although one technique requires that the algorithm is run a second time). Our probability of interest is still π_ψ, the probability of entering some rare goal set before a more typical taboo set, starting from the initial state x_0. The main motivation for the algorithm of Chapter 5 was efficient simulation of π_ψ using Zero Variance Approximation (ZVA), where the approximation function was given by $w^\diamond(x)$, the sum of the dominant path probabilities from x to the goal set. Inspired by [59], we explore in this chapter the idea of using the information to obtain $w^\diamond$ to remove the estimator variance resulting from the estimation of the dominant path probabilities. As we will explain in this chapter, this idea can also be applied if we use a change of measure that is not inspired by ZVA — e.g., one inspired by BFB.

The outline of this chapter is as follows: in Section 6.1 we introduce *two* methods that can be applied to improve the efficiency of simulation methods: the first method uses only the knowledge of the dominant paths obtained using the algorithm of Chapter 5, while the second method requires that this algorithm is run a second time. In Section 6.2 we analyse the performance of these methods. We demonstrate the findings of Section 6.2 using an illustrative example in Section 6.3 and using simulation results in Section 6.4, both for ZVA and for BFB. Section 6.5 concludes the chapter.

6.1 The Two New Methods

Using the algorithm outlined in Chapter 5, one obtains a new measure $\mathbb{Q}$ that implies an estimator with vanishing relative error. While this is a very desirable property for an estimator, we can further improve on this by using quantities computed explicitly while running the algorithm. Note that this method can be applied for any $\mathbb{Q}$, not just the change of measure of ZVA. As we will see, it requires 1) that we can identify, given a path, whether this path is dominant and 2) exact knowledge of the sum of the dominant path probabilities. It is not required that we actually use this knowledge for the change of measure.

Throughout this chapter we will use the following shorthand notation: Ω_ψ for $\Omega_\psi(x_0)$ as defined in (5.1) and Δ for the set of dominant paths from x_0, i.e., $\Delta = \Omega_\psi^{d(x_0)}(x_0)$, and

$$\mathbb{P}(\Delta) = w^\diamond(x_0) = \sum_{\omega \in \Delta} \mathbb{P}(\omega).$$

Recall that we use the short-hand notation $\mathbb{P}_x(\psi) = \mathbb{P}_x(\{\omega : \omega \vDash \psi\})$. Writing $\mathbb{P}(\psi)$ for $\mathbb{P}_{x_0}(\psi)$ and observing that $\Delta \subset \Omega_\psi$, we obtain the following two expressions for this quantity:

$$\begin{aligned} \mathbb{P}(\psi) &= \mathbb{P}(\Omega_\psi \cap \Delta) + \mathbb{P}(\Omega_\psi \cap \neg\Delta) \\ &= \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{P}}(\mathbf{1}_\psi \cdot \mathbf{1}_{\neg\Delta}) \\ &= \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi \cdot \mathbf{1}_{\neg\Delta}) \end{aligned} \tag{6.1}$$

and

$$\begin{aligned} \mathbb{P}(\psi) &= \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi \cdot \mathbf{1}_{\neg\Delta}) \\ &= \mathbb{P}(\Delta) + \mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi | \neg\Delta)(1 - \mathbb{Q}(\Delta)). \end{aligned} \tag{6.2}$$

Expression (6.1) contains $\mathbb{P}(\Delta)$, while (6.2) contains both $\mathbb{P}(\Delta)$ and $\mathbb{Q}(\Delta)$. The main insight is then that the exact value of $\mathbb{P}(\Delta)$ can be or has been computed using the algorithm of Chapter 5, while $\mathbb{Q}(\Delta)$ can be obtained by running the second phase — i.e., Algorithm 5.3 — a second time, but under the new measure $\mathbb{Q}$ instead of the original measure. We can use these observations to achieve variance reduction: if we were to estimate $\mathbb{P}(\psi)$ directly using (1.9), we would also incur variance through estimating $\mathbb{P}(\Delta)$ and $\mathbb{Q}(\Delta)$. However, when these quantities are known, they can be used directly. Hence, we propose the following two estimation procedures, based on ideas presented in [59].

The **first procedure** — which we call the $+$-method — is based on (6.1): we use knowledge of $\mathbb{P}(\Delta)$ and estimate the probability contribution of the remainder. To achieve this, we simulate under the new measure $\mathbb{Q}$, and check during the ith run of the procedure if sample path ω_i was part of Δ. If so, we set the corresponding term in the estimator to zero. Specifically, after having sampled N realisations $\omega_1, \ldots, \omega_N$, we use the following estimator for $\mathbb{P}(\psi)$:

$$\hat{\pi}_\psi^+ \triangleq \mathbb{P}(\Delta) + \frac{1}{N} \sum_{i=1}^{N} L_{\mathbb{Q}}(\omega_i) \cdot \mathbf{1}_\psi(\omega_i) \cdot \mathbf{1}_{\neg\Delta}(\omega_i). \tag{6.3}$$

Since all the sample paths are pairwise independent, the variance of the estimator under $\mathbb{Q}$ is given by

$$\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_\psi^+) = \mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi \cdot \mathbf{1}_{\neg\Delta}). \tag{6.4}$$

Writing $y_i = L_{\mathbb{Q}}(\omega_i) \cdot \mathbf{1}_\psi(\omega_i) \cdot \mathbf{1}_{\neg\Delta}(\omega_i)$, the variance of the estimator is estimated by

$$\hat{\sigma}_{\mathbb{Q}}^{2+} \triangleq \frac{\frac{1}{N}\sum_{i=1}^{N} y_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} y_i\right)^2}{N-1},$$

which gives rise to the approximate 95%-confidence interval

$$\left[\hat{\pi}_\psi^+ - 1.96\sqrt{\hat{\sigma}_{\mathbb{Q}}^{2+}}, \hat{\pi}_\psi^+ + 1.96\sqrt{\hat{\sigma}_{\mathbb{Q}}^{2+}}\right]. \tag{6.5}$$

The **second procedure** — which we call the $++$-method — is based on (6.2) and uses knowledge of both $\mathbb{P}(\Delta)$ and $\mathbb{Q}(\Delta)$. We again generate samples $\omega_1, \ldots, \omega_N$, but if ω_i is in Δ we *discard* it, giving rise to the alternative sample $\omega_1', \ldots, \omega_M'$ where M is the number of samples that are not in Δ. The reason is that by doing so, we remove the variance resulting from estimating $\mathbb{P}(\Omega_\psi \cap \neg\Delta)$. We then use the following estimator for $\mathbb{P}(\psi)$:

$$\hat{\pi}_\psi^{++} \triangleq \mathbb{P}(\Delta) + \frac{1 - \mathbb{Q}(\Delta)}{M} \sum_{i=1}^{M} L_{\mathbb{Q}}(\omega_i') \cdot \mathbf{1}_\psi(\omega_i'). \tag{6.6}$$

The variance of this estimator under $\mathbb{Q}$ is given by

$$\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_\psi^{++}) = \frac{(1 - \mathbb{Q}(\Delta))^2}{M} \mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi | \neg\Delta). \tag{6.7}$$

With $u_i = L_{\mathbb{Q}}(\omega_i') \cdot \mathbf{1}_\psi(\omega_i')$, the variance of this estimator is estimated by

$$\hat{\sigma}_{\mathbb{Q}}^{2++} \triangleq (1 - \mathbb{Q}(\Delta))^2 \, \frac{\frac{1}{M}\sum_{i=1}^{M} u_i^2 - \left(\frac{1}{M}\sum_{i=1}^{M} u_i\right)^2}{M - 1},$$

which in turn gives rise to the approximate 95%-confidence interval

$$\left[\hat{p}_{\mathbb{Q}}^{++} - 1.96\sqrt{\hat{\sigma}_{\mathbb{Q}}^{2++}}, \hat{p}_{\mathbb{Q}}^{++} + 1.96\sqrt{\hat{\sigma}_{\mathbb{Q}}^{2++}}\right].$$

In the next section, we compare the performance of these methods.

6.2 Performance of the Methods

In summary, we now have three estimators for π_ψ, namely the original estimator $\hat{\pi}_\psi$ (as defined in (1.9)), $\hat{\pi}_\psi^+$ (as defined in (6.3)) and $\hat{\pi}_\psi^{++}$ (as defined in (6.6)). The main question is then of course which one should be used in which situation. We answer this question by pairwise comparison of the estimators in the next three sections.

6.2.1 Comparing $\hat{\pi}_\psi$ to $\hat{\pi}_\psi^+$

We first compare $\hat{\pi}_\psi$ to $\hat{\pi}_\psi^+$. Note that we have

$$\begin{aligned} \mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_\psi) &= \frac{1}{N}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi) = \frac{1}{N}\left(\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^2 \cdot \mathbf{1}_\psi) - \mathbb{E}_{\mathbb{Q}}^2(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi)\right) \\ &= \frac{1}{N}\left(\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}} \cdot \mathbf{1}_\psi) - \mathbb{P}^2(\psi)\right). \end{aligned} \tag{6.8}$$

Analogously,

$$\begin{aligned}\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_{\psi}^{+}) &= \frac{1}{N}\left(\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^{2}\cdot\mathbf{1}_{\psi}\cdot\mathbf{1}_{\neg\Delta}) - \mathbb{E}_{\mathbb{Q}}^{2}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}\cdot\mathbf{1}_{\neg\Delta})\right)\\ &= \frac{1}{N}\left(\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}\cdot\mathbf{1}_{\neg\Delta}) - \mathbb{P}^{2}(\Omega_{\psi}\cap\neg\Delta)\right).\end{aligned} \tag{6.9}$$

Combining the two, we obtain

$$\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_{\psi}) - \mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_{\psi}^{+}) = \frac{1}{N}\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\Delta}) - \frac{1}{N}\left[\mathbb{P}^{2}(\psi) - \mathbb{P}^{2}(\Omega_{\psi}\cap\neg\Delta)\right].$$

Both terms in this expression are positive, but whether the result is positive or negative depends on $\mathbb{Q}$. This is demonstrated in Sections 6.3 and 6.4, in which it turns out that the $+$-method of (6.3) is better for BFB and the standard estimator is better for ZVA for the chosen examples.

6.2.2 Comparing $\hat{\pi}_{\psi}$ to $\hat{\pi}_{\psi}^{++}$

As for the comparison between $\hat{\pi}_{\psi}$ and $\hat{\pi}_{\psi}^{++}$, note that we can write

$$\begin{aligned}\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_{\psi}) &= \frac{1}{N}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi})\\ &= \frac{1}{N}\left(\mathbb{E}_{\mathbb{Q}}[\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\mathbf{1}_{\Delta})] + \mathrm{Var}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\mathbf{1}_{\Delta})]\right)\\ &= \frac{1}{N}\mathbb{Q}(\neg\Delta)\cdot\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\neg\Delta) + \frac{1}{N}\mathbb{Q}(\Delta)\cdot\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\Delta)\\ &\quad + \frac{1}{N}\mathrm{Var}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\mathbf{1}_{\Delta})],\end{aligned} \tag{6.10}$$

where the second equality follows from the law of total variance. Since $M \approx N\cdot\mathbb{Q}(\neg\Delta)$, we have that

$$\frac{1}{N}\mathbb{Q}(\neg\Delta)\cdot\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\neg\Delta) \approx \frac{(1-\mathbb{Q}(\Delta))^{2}}{M}\cdot\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_{\psi}|\neg\Delta),$$

which is exactly $\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_{\psi}^{++})$ by (6.7). Since the other terms in the final expression of (6.10) are positive (variances are always positive), $\hat{\pi}_{\psi}^{++}$ will have smaller variance than $\hat{\pi}_{\psi}$ as long as the approximation $M \approx N\cdot\mathbb{Q}(\neg\Delta)$ is justifiable.

6.2.3 Comparing $\hat{\pi}_\psi^+$ to $\hat{\pi}_\psi^{++}$

Finally, we compare $\hat{\pi}_\psi^+$ and $\hat{\pi}_\psi^{++}$. To this effect, we write the variance of $\hat{\pi}_\psi^{++}$ as

$$\begin{aligned}
\mathrm{Var}_{\mathbb{Q}}(\hat{\pi}_\psi^{++}) &= \frac{\mathbb{Q}^2(\neg\Delta)}{M}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_\psi|\neg\Delta) \\
&\approx \frac{\mathbb{Q}(\neg\Delta)}{N}\mathrm{Var}_{\mathbb{Q}}(L_{\mathbb{Q}}\cdot\mathbf{1}_\psi|\neg\Delta) \\
&= \frac{\mathbb{Q}(\neg\Delta)}{N}\left[\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^2\mathbf{1}_\psi|\neg\Delta) - \mathbb{E}_{\mathbb{Q}}^2(L_{\mathbb{Q}}\mathbf{1}_\psi|\neg\Delta)\right] \\
&= \frac{\mathbb{Q}(\neg\Delta)}{N}\left[\frac{\mathbb{E}_{\mathbb{Q}}(L_{\mathbb{Q}}^2\mathbf{1}_\psi\mathbf{1}_{\neg\Delta})}{\mathbb{Q}(\neg\Delta)} - \frac{\mathbb{E}_{\mathbb{Q}}^2(L_{\mathbb{Q}}\mathbf{1}_\psi\mathbf{1}_{\neg\Delta})}{\mathbb{Q}^2(\neg\Delta)}\right] \\
&= \frac{1}{N}\left(\mathbb{E}_{\mathbb{P}}(L_{\mathbb{Q}}\cdot\mathbf{1}_\psi\cdot\mathbf{1}_{\neg\Delta}) - \frac{\mathbb{P}^2(\Omega_\psi\cap\neg\Delta)}{\mathbb{Q}(\neg\Delta)}\right).
\end{aligned} \tag{6.11}$$

This expression is easily compared with (6.9), and since $\mathbb{Q}(\neg\Delta)$ is smaller than 1 it holds that the variance of $\hat{\pi}_\psi^{++}$ is lower than the variance of $\hat{\pi}_\psi^+$.

In the numerical results of Section 6.4 one indeed observes that $\hat{\pi}_\psi^{++}$ performs better than the standard estimator and $\hat{\pi}_\psi^+$. However, in the setting of Chapter 5 we typically have that $\mathbb{Q}(\Delta)\to 1$ as $\epsilon\downarrow 0$, which means that observing a non-dominant path becomes a rare event in itself. This can be a problem because we cannot construct a confidence interval for $\hat{\pi}_\psi^{++}$ if we do not observe any non-dominant paths. Furthermore, if M is small the approximation $M\approx\mathbb{Q}(\neg\Delta)N$ is no longer accurate. The effects of this will be demonstrated empirically in Section 6.4.2. Before that, we aim to develop intuition for the quantities introduced in this chapter — this is done in the following section.

6.3 Illustrative Example

In this section we illustrate the analysis of the preceding sections using a simple example, namely the example given in Figure 6.1(a). It represents a very simple birth-death process, as discussed in Chapter 3. Births occur with rate ϵ, regardless of the population size, and organisms die with rate $1-\epsilon$. Using the algorithm of Chapter 5 we obtain

$$d(x)=\begin{cases}\infty & \text{for } x=x_a,\\ 2 & \text{for } x=x_0,\\ 1 & \text{for } x=x_1,\\ 0 & \text{for } x=x_b\end{cases}\quad\text{and } w(x)=\begin{cases}0 & \text{for } x=x_a,\\ \epsilon^2 & \text{for } x=x_0,\\ \epsilon & \text{for } x=x_1,\\ 1 & \text{for } x=x_b.\end{cases}$$

The set of paths in this setting has a very simple form: paths either end up in x_a or x_b and while doing so take the transition from x_1 back to x_0 exactly i times $\forall i\in\mathbb{N}$.

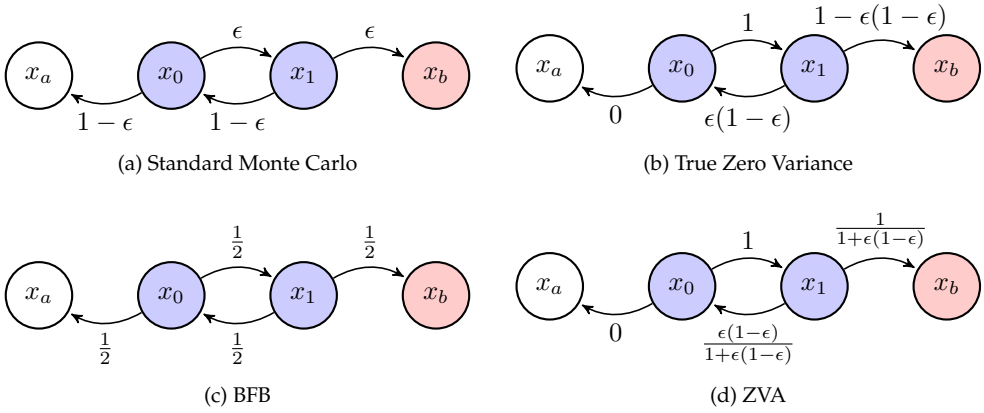
The probabilities of interest can be easily expressed in closed form:

$$\mathbb{P}_{x_0}(\psi) = \sum_{\omega \vDash \psi} \mathbb{P}_{x_0}(\omega) = \epsilon^2 \sum_{i=0}^{\infty} (\epsilon(1-\epsilon))^i = \frac{\epsilon^2}{1-\epsilon(1-\epsilon)}$$

and, similarly, $\mathbb{P}_{x_1}(\psi) = \frac{\epsilon}{1-\epsilon(1-\epsilon)}$. Using this information, the true zero variance measure of (1.12) can be determined; the result is displayed in Figure 6.1(b).

In Figures 6.1(c) and (d) we depict the simulation measures implied by the two main IS methods considered in this thesis: Balanced Failure Biasing (BFB) as described in Section 1.3.4 (BFB) and the approach based on Zero Variance Approximation (ZVA) as described in Chapter 5. To apply BFB we need a clear distinction between failures and repairs; since the example corresponds to a birth-death process this distinction is trivial. For ZVA we use the change of measure given by (5.5) combined with the knowledge of $w^\diamond$ given previously. The simulation measures for BFB and ZVA will be denoted by $\mathbb{Q}_{\text{BFB}}$ and $\mathbb{Q}_{\text{ZVA}}$ respectively; whenever $\mathbb{Q}$ is used as a subscript we use the name of the method for brevity; e.g., we write Var_{BFB} instead of $\text{Var}_{\mathbb{Q}_{\text{BFB}}}$. Note that for a path ω that takes the transition from x_1 back to x_0 exactly i times, we have

$$\begin{aligned}
&\mathbb{P}(\omega) = \epsilon^{2+i}(1-\epsilon)^i, \\
&\mathbb{Q}_{\text{BFB}}(\omega) = \frac{1}{4^{1+i}}, && \mathbb{Q}_{\text{ZVA}}(\omega) = \frac{\epsilon^i(1-\epsilon)^i}{(1+\epsilon(1-\epsilon))^{1+2i}}, \\
&L_{\text{BFB}}(\omega) = 4\epsilon^2 \cdot (4\epsilon(1-\epsilon))^i, \quad \text{and} && L_{\text{ZVA}}(\omega) = \frac{\epsilon^2}{(1+\epsilon(1-\epsilon))^{1+2i}}.
\end{aligned}$$



(a) Standard Monte Carlo

(b) True Zero Variance

(c) BFB

(d) ZVA

Figure 6.1: The illustrating example under four probability measures: (a) the original measure $\mathbb{P}$, (b) the true zero variance measure, (c) the IS-measure implied by BFB and (d) the IS-measure implied by ZVA as described in Chapter 5.

We first discuss the performance of BFB. By (6.8) we can write the variance of the standard estimator as

$$\begin{aligned} N \cdot \mathrm{Var}_{\mathrm{BFB}}(\hat{\pi}_\psi) &= \sum_{\omega \in \psi} L_{\mathrm{BFB}}(\omega)\mathbb{P}(\omega) - \mathbb{P}^2(\psi) \\ &= \left(4\epsilon^4 \sum_{i=0}^{\infty} 4^i \cdot (\epsilon(1-\epsilon))^{2i}\right) - \left(\frac{\epsilon^2}{1-\epsilon(1-\epsilon)}\right)^2 \\ &= \frac{\epsilon^4(3 - 8\epsilon + 16\epsilon^2 - 16\epsilon^3 + 8\epsilon^4)}{(1 - 4\epsilon^2 + 8\epsilon^3 - 4\epsilon^4)(1-\epsilon(1-\epsilon))^2}. \end{aligned}$$

The left term and right term in the third expression are both $\Theta(\epsilon^4)$, but they do not cancel out so that the resulting expression is also $\Theta(\epsilon^4)$. For the estimator $\hat{\pi}_\psi^+$ under BFB we have by (6.9) that

$$\begin{aligned} N \cdot \mathrm{Var}_{\mathrm{BFB}}(\hat{\pi}_\psi^+) &= \sum_{\omega \in \psi \wedge \neg\Delta} L_{\mathrm{BFB}}(\omega)\mathbb{P}(\omega) - \mathbb{P}^2(\Omega_\psi \cap \neg\Delta) \\ &= \left(\epsilon^4 \sum_{i=1}^{\infty} 4^i \cdot (\epsilon(1-\epsilon))^{2i}\right) - \left(\frac{\epsilon^3(1-\epsilon)}{1-\epsilon(1-\epsilon)}\right)^2 \\ &= \frac{\epsilon^6(1-\epsilon)^2(63 - 128\epsilon + 208\epsilon^2 - 160\epsilon^3 + 80\epsilon^4)}{(1 - 16\epsilon^2 + 32\epsilon^3 - 16\epsilon^4)(1-\epsilon(1-\epsilon))^2}. \end{aligned}$$

Here, both terms in the third expression are $\Theta(\epsilon^6)$, and because they do not cancel out the total variance is also $\Theta(\epsilon^6)$. For the estimator $\hat{\pi}_\psi^{++}$ under BFB, note that $\mathbb{Q}_{\mathrm{BFB}}(\Delta) = \frac{1}{4}$, so that by (6.11) we have that

$$\begin{aligned} N \cdot \mathrm{Var}_{\mathrm{BFB}}(\hat{\pi}_\psi^{++}) &= \sum_{\omega \in \psi \wedge \neg\Delta} L_{\mathrm{BFB}}(\omega)\mathbb{P}(\omega) - \frac{\mathbb{P}^2(\Omega_\psi \cap \neg\Delta)}{\mathbb{Q}_{\mathrm{BFB}}(\neg\Delta)} \\ &= \left(4\epsilon^4 \sum_{i=1}^{\infty} 4^i \cdot (\epsilon(1-\epsilon))^{2i}\right) - \frac{4}{3}\left(\frac{\epsilon^3(1-\epsilon)}{1-\epsilon(1-\epsilon)}\right)^2 \\ &= \frac{4}{3}\frac{\epsilon^6(1-\epsilon)^2(47 - 96\epsilon + 160\epsilon^2 - 128\epsilon^3 + 64\epsilon^4)}{(1 - 16\epsilon^2 + 32\epsilon^3 - 16\epsilon^4)(1-\epsilon(1-\epsilon))^2}. \end{aligned}$$

Again, both terms in the third expression are $\Theta(\epsilon^6)$ and because they do not cancel out the result is also $\Theta(\epsilon^6)$. In conclusion, the difference between $\hat{\pi}_\psi$ and the new methods (in terms of variance) is of two ϵ-orders, while the difference between $\hat{\pi}_\psi^+$ and $\hat{\pi}_\psi^{++}$ is relatively small.

For ZVA, the variance of the standard estimator is, again by (6.8),

$$\begin{aligned} N \cdot \mathrm{Var}_{\mathrm{ZVA}}(\hat{\pi}_\psi) &= \sum_{\omega \in \psi} L_{\mathrm{ZVA}}(\omega)\mathbb{P}(\omega) - \mathbb{P}^2(\psi) \\ &= (1+\epsilon(1-\epsilon)) \cdot \epsilon^4 \sum_{i=0}^{N} \left[\left(\frac{1+\epsilon(1-\epsilon)}{\epsilon(1-\epsilon)} \right)^i \cdot (\epsilon(1-\epsilon))^{2i} \right] - \left(\frac{\epsilon^2}{1-\epsilon(1-\epsilon)} \right)^2 \\ &= \frac{\epsilon^7 \cdot (1 - 3\epsilon + 3\epsilon^2 - \epsilon^3)}{(1-\epsilon(1-\epsilon))^2 \cdot (1 - \epsilon + 2\epsilon^3 - \epsilon^4)}. \end{aligned}$$

Both terms in the third expression are $\Theta(\epsilon^4)$; however, their difference is $\Theta(\epsilon^7)$, which is much lower. For the estimator $\hat{\pi}^+_\psi$ under ZVA we have by (6.9) that

$$\begin{aligned} N \cdot \mathrm{Var}_{\mathrm{ZVA}}(\hat{\pi}^+_\psi) &= \sum_{\omega \in \psi \wedge \neg\Delta} L_{\mathrm{ZVA}}(\omega)\mathbb{P}(\omega) - \mathbb{P}^2(\Omega_\psi \cap \neg\Delta) \\ &= (1+\epsilon(1-\epsilon)) \cdot \epsilon^4 \sum_{i=1}^{\infty} \left[\left(\frac{1+\epsilon(1-\epsilon)}{\epsilon(1-\epsilon)} \right)^i \cdot (\epsilon(1-\epsilon))^{2i} \right] - \left(\frac{\epsilon^3(1-\epsilon)}{1-\epsilon(1-\epsilon)} \right)^2. \end{aligned}$$

Here, the left term is $\Theta(\epsilon^5)$ and the right term is $\Theta(\epsilon^6)$, resulting in a total variance of $\Theta(\epsilon^5)$. For the estimator $\hat{\pi}^{++}_\psi$ under ZVA we have $\mathbb{Q}_{\mathrm{ZVA}}(\Delta) = \frac{1}{1+\epsilon(1-\epsilon)}$, so that

$$\begin{aligned} N \cdot \mathrm{Var}_{\mathrm{ZVA}}(\hat{\pi}^{++}_\psi) &= \sum_{\omega \in \psi \wedge \neg\Delta} L_{\mathrm{ZVA}}(\omega)\mathbb{P}(\omega) - \frac{\mathbb{P}^2(\Omega_\psi \cap \neg\Delta)}{\mathbb{Q}_{\mathrm{ZVA}}(\neg\Delta)} \\ &= (1+\epsilon(1-\epsilon)) \cdot \epsilon^4 \sum_{i=1}^{\infty} \left[\left(\frac{1+\epsilon(1-\epsilon)}{\epsilon(1-\epsilon)} \right)^i \cdot (\epsilon(1-\epsilon))^{2i} \right] \\ &\quad - \left(\frac{\epsilon^3(1-\epsilon)}{1-\epsilon(1-\epsilon)} \right)^2 \cdot \left(1 - \frac{1}{1+\epsilon(1-\epsilon)} \right)^{-1} \\ &= \frac{\epsilon^8 \cdot (1 - 3\epsilon + 3\epsilon^2 - \epsilon^3) \cdot (1+\epsilon(1-\epsilon)) \cdot (1-\epsilon)}{(1-\epsilon(1-\epsilon))^2 \cdot (1 - \epsilon + 2\epsilon^3 - \epsilon^4)}. \end{aligned}$$

by (6.11). Both terms in the third expression are $\Theta(\epsilon^5)$, but their difference is $\Theta(\epsilon^8)$. So for ZVA it holds that $\hat{\pi}^+_\psi$ performs several ϵ-orders worse than $\hat{\pi}_\psi$, but $\hat{\pi}^{++}_\psi$ performs an ϵ-order better than $\hat{\pi}_\psi$.

The final method that we consider is standard Monte Carlo — its variance is given by

$$N \cdot \mathrm{Var}_{\mathbb{P}}(\hat{\pi}_\psi) = \mathbb{P}(\psi) - \mathbb{P}^2(\psi).$$

The left term is $\Theta(\epsilon^2)$ and the second term is $\Theta(\epsilon^4)$, meaning that the result is $\Theta(\epsilon^2)$.

In Table 6.1, we display a summary of the results of this section. In Section 6.4 we will observe that the empirical results match the general picture drawn in that table. The table suggests that if information about the dominant paths has been obtained through the algorithm of Chapter 5, it is best to use ZVA instead of BFB.

method	origin	MC	BFB	ZVA
normal	(1.9)	$\Theta(\epsilon^4)$	$\Theta(\epsilon^2)$	$\Theta(\epsilon^7)$
$+$	(6.3)		$\Theta(\epsilon^6)$	$\Theta(\epsilon^5)$
$++$	(6.6)		$\Theta(\epsilon^6)$	$\Theta(\epsilon^8)$

Table 6.1: Variances of the methods discussed in this chapter.

Furthermore, among the ZVA-methods it holds that ZVA^{++} is the best. Note, however, that $\mathbb{Q}(\neg\Delta) = (\epsilon(1-\epsilon))/(1+\epsilon(1-\epsilon)) = \Theta(\epsilon)$, so that the probability of observing a non-dominant path vanishes as $\epsilon \downarrow 0$. If only dominant paths have been observed in a simulation run (i.e., $M = 0$), then the ZVA^{++}-estimate of π_ψ would be $\mathbb{P}(\Delta) = \epsilon^2$ without a well-defined confidence interval. One can compare this with ZVA, for which the estimate is then $\epsilon^2/(1+\epsilon(1+\epsilon))$ with an estimated variance of 0. The confidence interval is now deceptively narrow (that is, the true value of π_ψ is not contained in the CI), but well-defined. More importantly, some information about the non-dominant paths is included in the ZVA-estimate, as we discuss below.

To see why the non-dominant paths affect the ZVA-estimate, note that $w(x_0) = p_{x_0x_1} \cdot p_{x_1x_b}$ and that $w(x_b) = 1$, so that the ZVA-estimate based on only the dominant path ω^* is given by

$$\begin{aligned}\frac{p_{x_0x_1} \cdot p_{x_1x_b}}{q_{x_0x_1} \cdot q_{x_1x_b}} &= p_{x_0x_1} \cdot p_{x_1x_b} \cdot \frac{p_{x_1x_0}w(x_0) + p_{x_1x_b}w(x_b)}{p_{x_1x_b}w(x_b)} \\ &= p_{x_0x_1} \cdot p_{x_1x_0} \cdot p_{x_0x_1} \cdot p_{x_1x_b} + p_{x_0x_1} \cdot p_{x_1x_b},\end{aligned}$$

which is exactly the sum of the probability of the dominant path and the path that falls back once to x_0. The intuition of why this happens is that the probability under $\mathbb{Q}_{\text{ZVA}}$ of jumping from x_1 to x_b is computed by comparing the jumps to x_0 and x_b in terms of probability contribution to π_ψ. To make this comparison, the probability of the path that falls back to x_0 and then goes straight to x_b is computed and accounted for in the likelihood ratio. The result is an estimate that is closer to π_ψ than $\mathbb{P}(\Delta)$ (the ZVA^{++}-estimate) is. Hence, the ZVA-estimate is in several ways superior to ZVA^{++}-estimate when M is very small.

6.4 Numerical Results

In this section we will investigate empirically the efficiency of the methods introduced in Section 6.1. We will consider two model settings: the illustrative example of Section 6.3 and an artificial model containing a high-probability cycle. All of these models are small and (intended to be) intuitively clear, consisting of up to seven states at most.

For each model, we first determine d and $w^\diamond$ evaluated for all elements of the state space and display these values in Tables 6.2 and 6.4. We then display the

results of simulation experiments for several values of ϵ in Tables 6.3 and 6.5. In each of the tables dedicated to the experiments, we compare a number of simulation approaches. These are standard Monte Carlo simulation (MC), the importance sampling estimator using BFB and using ZVA as described in Chapter 5. For both of the IS-procedures we use three versions, the standard version, the $+$-version in which we use knowledge of $\mathbb{P}(\Delta)$ and the $++$-version in which we use knowledge of both $\mathbb{P}(\Delta)$ and $\mathbb{Q}(\Delta)$. For all these cases, we display a 95%-confidence interval for $\hat{\pi}_\psi$ based on the central limit theorem, the total number of simulated runs N and the number of runs M in which a non-dominant path was sampled (i.e., a path ω such that $\mathbb{P}(\omega) \neq \Theta(\epsilon^{d(x_0)})$). The run time of each simulation experiment was set to one second.

6.4.1 Illustrative Example

We first attempt to empirically demonstrate the findings of Section 6.3. As such, we consider the DTMC displayed in Figure 6.1(a). From each state there is only a single dominant path that leads to x_b, which explains the simple structure of $w^\diamond$ in Table 6.2.

	x_0	x_1	x_2	x_3	x_b	x_a
d	3	3	2	1	0	∞
$w^\diamond$	ϵ^3	ϵ^3	ϵ^2	ϵ	1	0

Table 6.2: Functions $w^\diamond$ and d for the example of Figure 6.1(a).

In Table 6.3, we display the simulation results for this model. As we can see, the results largely match the results of Table 6.1 although the effect of the hidden constants[24] is clearly visible. The importance sampling estimators appear to be unbiased and clearly outperform the standard MC estimator for small values of ϵ, although for small values of ϵ we may run into numerical problems. This is especially visible for the ZVA^{++}-estimate for $\epsilon = 10^{-4}$, for which the confidence interval does not contain the true value $\epsilon^2/(1-\epsilon+\epsilon^2)$. This is because a path that falls back to x_0 once should have a likelihood ratio equal to $1.000200030 \cdot 10^{-8}$, while the way we compute this number in Java yields the floating point number $1.0001999899969999 \cdot 10^{-8}$ due to numerical imprecision.

In this model, every time a dominant path ω is sampled, the likelihood ratio $L_{\mathbb{Q}}(\omega)$ will be the same, so there is not much variance from estimating $\mathbb{P}(\Delta)$. Still, ZVA++ is noticeably better than ZVA. We also see that ZVA+ is even worse than normal ZVA; this is because the true probability is much closer to the likelihood ratios of samples in which the dominant path was observed than to the value 0 which is used instead. On the other hand, BFB+ and BFB++ are not very different in terms of estimator variance, and both clearly outperform BFB. Also, ZVA still outperforms BFB++.

[24]The hidden constant of a function f of ϵ that is $\Theta(\epsilon^k)$ is the $c \in \mathbb{R}^+$ for which $\lim_{\epsilon \downarrow 0} \epsilon^{-k} f(\epsilon) = c$.

ϵ	method	$\hat{\pi}$	CI-bounds	N	M
10^{-1}	MC	$1.09984 \cdot 10^{-2}$	$\pm\, 3.79 \cdot 10^{-5}$	29 098 592	28 807 138
	BFB	$1.098105 \cdot 10^{-2}$	$\pm\, 8.37 \cdot 10^{-6}$	16 033 109	12 027 841
	BFB+	$1.098999 \cdot 10^{-2}$	$\pm\, 1.73 \cdot 10^{-6}$	15 904 766	11 930 959
	BFB++	$1.098974 \cdot 10^{-2}$	$\pm\, 1.48 \cdot 10^{-6}$	15 832 011	11 875 091
	ZVA	$1.0989055 \cdot 10^{-2}$	$\pm\, 1.39 \cdot 10^{-7}$	19 316 924	1 595 178
	ZVA+	$1.098915 \cdot 10^{-2}$	$\pm\, 1.44 \cdot 10^{-6}$	20 051 867	1 655 861
	ZVA++	$1.09890187 \cdot 10^{-2}$	$\pm\, 1.23 \cdot 10^{-8}$	20 016 407	1 652 133
10^{-3}	MC	$1.068 \cdot 10^{-6}$	$\pm\, 3.44 \cdot 10^{-7}$	34 652 217	34 652 180
	BFB	$1.000806 \cdot 10^{-6}$	$\pm\, 8.6 \cdot 10^{-10}$	15 564 650	11 674 251
	BFB+	$1.00099998 \cdot 10^{-6}$	$\pm\, 1.9 \cdot 10^{-12}$	15 944 632	11 957 412
	BFB++	$1.00099936 \cdot 10^{-6}$	$\pm\, 1.68 \cdot 10^{-12}$	14 966 588	11 225 985
	ZVA	$1.0010000087 \cdot 10^{-6}$	$\pm\, 1.88 \cdot 10^{-14}$	21 125 016	21 297
	ZVA+	$1.0010064 \cdot 10^{-6}$	$\pm\, 1.32 \cdot 10^{-11}$	22 351 332	22 450
	ZVA++	$1.0009999989795 \cdot 10^{-6}$	$\pm\, 1.3 \cdot 10^{-17}$	22 225 832	22 461
10^{-4}	MC	$3.05 \cdot 10^{-8}$	$\pm\, 5.98 \cdot 10^{-8}$	32 760 810	32 760 809
	BFB	$1.000067 \cdot 10^{-8}$	$\pm\, 8.68 \cdot 10^{-12}$	15 296 995	11 472 874
	BFB+	$1.000100054 \cdot 10^{-8}$	$\pm\, 1.96 \cdot 10^{-15}$	15 067 059	11 303 922
	BFB++	$1.000099909 \cdot 10^{-8}$	$\pm\, 1.67 \cdot 10^{-15}$	15 223 434	11 418 247
	ZVA	$1.0000999996 \cdot 10^{-8}$	$\pm\, 1.05 \cdot 10^{-16}$	20 420 787	2 020
	ZVA+	$1.00009767 \cdot 10^{-8}$	$\pm\, 4.26 \cdot 10^{-14}$	20 716 099	2 023
	ZVA++	$1.0001000000028801 \cdot 10^{-8}$	$\pm\, 9.71 \cdot 10^{-22}$	20 245 335	2 049

Table 6.3: Simulation results for example of Figure 6.1(a).

6.4.2 Two Path Model

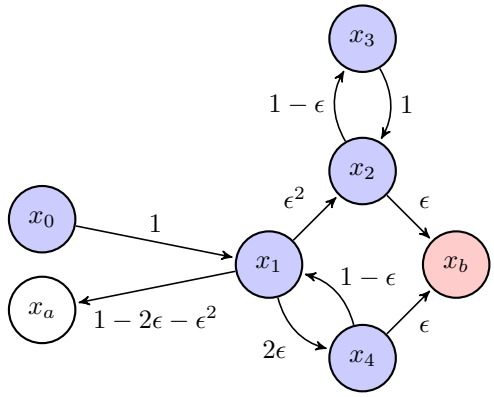


Figure 6.2: The model with two paths as discussed in Section 6.4.2, under the original transition probability matrix P.

The model of Figure 6.2 is simple, but we use it to demonstrate both the findings of Section 6.3 and the functioning of our algorithm in the context of high-probability cycles (as discussed in the introduction to Chapter 5). Note that the DTMC consisting of only x_a, x_1, x_4 and x_b and the transitions between these states is equivalent to the model of Figure 6.1 with slightly different rates. The paths to x_b that leave this subchain are of at least the same ϵ-orders, so the behaviour of the variances of the estimators should behave as in Table 6.1. Furthermore, since there is a high-probability cycle from x_2 to x_3, the approach of [73] does not work in this situation (it would assign too much probability mass to the paths going from x_1 to x_4). Using the procedure outlined in Chapter 5, we reduce the model of Figure 6.2 to a similar DTMC where the transition from x_2 to x_b and the transition from x_3 to x_b have been given probability 1. For this DTMC, the functions d and $w^\diamond$ are as displayed in Table 6.4, and the resulting DTMC is displayed in Figure 6.3. As in Chapter 5, we call the transition probability matrix of the new DTMC P'.

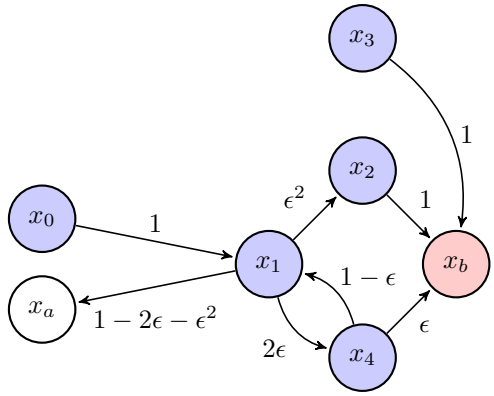


Figure 6.3: The model of Figure 6.2 under the new transition probability matrix P' — i.e., with the high-probability cycle between x_2 and x_3 removed.

Because there is a high-probability cycle between x_2 and x_3, BFB will not work well, as the resulting estimator will without necessity incur extra variance by changing the transition probabilities $p_{x_2x_3}$ and $p_{x_2x_b}$. A possible remedy is to apply the General Biasing Scheme as described in [60], which is specifically constructed to be able to deal with high-probability cycles. In this section, we simply instruct BFB to simulate under the old measure in state x_2 for the sake of a fair comparison.

The simulation results for this example are displayed in Table 6.5, about which some interesting observations can be made. First, we see that N, the number of paths sampled during the one second during which we ran the procedure, decreases as ϵ decreases; this is because we use the values from the unadjusted DTMC, and as ϵ decreases, the expected amount of time before the loop between x_2 and x_3 is left increases. This can be avoided if we use the probability values from P' to plug into (1.12) and the denominator of the likelihood ratio $L_{\mathbb{Q}}$.

Second, for BFB, the speed-up resulting from the analysis of Section 6.1 is again

	x_0	x_1	x_2	x_3	x_4	x_b	x_a
d	2	2	0	0	1	0	∞
$w^\diamond$	$3\epsilon^2$	$3\epsilon^2$	1	1	ϵ	1	0

Table 6.4: Functions $w^\diamond$ and d for the example of Figure 6.2.

ϵ	method	$\hat{\pi}$	CI-bounds	N	M
10^{-1}	MC	$3.65236 \cdot 10^{-2}$	$\pm\, 8.73 \cdot 10^{-5}$	17 745 411	17 213 873
	BFB	$3.6607 \cdot 10^{-2}$	$\pm\, 4.83 \cdot 10^{-5}$	5 799 350	3 623 894
	BFB+	$3.6577 \cdot 10^{-2}$	$\pm\, 2.82 \cdot 10^{-5}$	5 870 600	3 667 872
	BFB++	$3.65868 \cdot 10^{-2}$	$\pm\, 2.22 \cdot 10^{-5}$	5 800 283	3 625 487
	ZVA	$3.65864 \cdot 10^{-2}$	$\pm\, 5.87 \cdot 10^{-6}$	4 258 653	603 010
	ZVA+	$3.65838 \cdot 10^{-2}$	$\pm\, 1.51 \cdot 10^{-5}$	4 541 687	643 583
	ZVA++	$3.658572 \cdot 10^{-2}$	$\pm\, 1.04 \cdot 10^{-6}$	4 460 635	631 720
10^{-3}	MC	$3.65 \cdot 10^{-6}$	$\pm\, 7.05 \cdot 10^{-7}$	28 222 996	28 222 893
	BFB	$2.9951 \cdot 10^{-6}$	$\pm\, 3.37 \cdot 10^{-8}$	91 014	56 994
	BFB+	$3.006111 \cdot 10^{-6}$	$\pm\, 2.19 \cdot 10^{-10}$	90 620	56 859
	BFB++	$3.006069 \cdot 10^{-6}$	$\pm\, 1.71 \cdot 10^{-10}$	90 916	56 551
	ZVA	$3.0059895 \cdot 10^{-6}$	$\pm\, 2.96 \cdot 10^{-11}$	79 714	163
	ZVA+	$3.006026 \cdot 10^{-6}$	$\pm\, 9.51 \cdot 10^{-10}$	77 054	154
	ZVA++	$3.0060059764 \cdot 10^{-6}$	$\pm\, 6.13 \cdot 10^{-14}$	77 786	150
10^{-4}	MC	—	—	27 674 484	27 674 484
	BFB	$3.05 \cdot 10^{-8}$	$\pm\, 1.06 \cdot 10^{-9}$	9 279	5 725
	BFB+	$3.0005808 \cdot 10^{-8}$	$\pm\, 6.72 \cdot 10^{-13}$	8 914	5 507
	BFB++	$3.0005882 \cdot 10^{-8}$	$\pm\, 5.19 \cdot 10^{-13}$	9 531	6 060
	ZVA	$3.0005965 \cdot 10^{-8}$	$\pm\, 9.87 \cdot 10^{-14}$	7 151	3
	ZVA+	$3.00119 \cdot 10^{-8}$	$\pm\, 1.34 \cdot 10^{-11}$	7 577	3
	ZVA++	$3.00060002998 \cdot 10^{-8}$	$\pm\, 1.99 \cdot 10^{-17}$	7 894	2
10^{-5}	MC	—	—	28 046 932	28 046 932
	BFB	$3.083 \cdot 10^{-10}$	$\pm\, 3.59 \cdot 10^{-11}$	859	545
	BFB+	$3.0000831 \cdot 10^{-10}$	$\pm\, 2.62 \cdot 10^{-15}$	863	554
	BFB++	$3.0000515 \cdot 10^{-10}$	$\pm\, 1.67 \cdot 10^{-15}$	905	567
	ZVA	$3.00006037 \cdot 10^{-10}$	$\pm\, 3.06 \cdot 10^{-16}$	735	0
	ZVA+	$3.000 \cdot 10^{-10}$	$\pm\, 0$	753	0
	ZVA++	$3.000 \cdot 10^{-10}$	—	695	0

Table 6.5: Simulation results for example of Figure 6.2.

clearly visible. Third, we see that the methods of Section 6.1 lead to considerable variance reduction for ZVA when ϵ is moderately small (at $\epsilon = 10^{-3}$, there is a factor 20 improvement from ZVA to ZVA++), but when ϵ becomes so small that non-dominant paths are typically not sampled any longer (remember that we estimate the contribution of $\mathbb{E}_\mathbb{Q}(L_\mathbb{Q} \cdot \mathbf{1}_\psi | \neg\Delta)$ using a measure constructed for the dominant

paths), we often end up with (zero or) one sample(s), and no reasonable confidence interval can be given. Using the standard ZVA estimator, no non-dominant paths are sampled either, but some information about the non-dominant paths turns out to appear in the likelihood ratios (see the discussion at the end of Section 6.3), which means that the standard ZVA is even better than the one combined with the techniques of Section 6.1, where this information is thrown away in favour of the exact computation of $\mathbb{P}(\Delta)$.

6.5 Conclusions

We presented two new methods that can be used to achieve variance reduction for IS-methods by using information about the dominant paths: the $+$-method and the $++$-method. The $+$-method can be applied directly after running the algorithm of Chapter 5, whereas the $++$-method requires that it is run a second time. We demonstrated their potential gain (or loss), where the $++$-method stands out particularly positively. However, when the number of non-dominant paths sampled is very low, the original ZVA-estimator is to be preferred. Unfortunately, it may be difficult to say something about this beforehand.

CHAPTER 7

Dominant Paths in Stochastic Petri Nets

The topic of the previous four chapters has been importance sampling where the change of measure is based on the dominant paths to failure — in this chapter we focus on a method in which we obtain information about the dominant paths using a high-level description of the model. We are again interested in estimating π_ψ, the probability of entering a rare goal set before a more typical taboo set. As in Chapter 5, we assume that all the transition probabilities in our model depend on the rarity parameter ϵ, and we again use the ϵ-distance function d that was defined in (5.2) to obtain insight into which paths to the goal set are the most likely. Previously, we presented an algorithm to find d that worked at the level of the state space of the Markov chain. The state space of the Markov chain can be very large (even infinite), which may prove to be problematic for the Dijkstra-based algorithm of Chapter 5 as its runtime depends strongly on the size of the state space. The algorithm only needs to consider a subset of the state space, but even this subset may be too large for the algorithm to be feasible. In this chapter we use the fact that in practice the chain often has enough structure to allow for implicit specification using a high-level description language — information about the dominant paths to failure is often more easy to obtain from the high-level than from the low-level description. Classical examples of such high-level description languages are the Stochastic Petri Nets (SPNs) [2], Stochastic Activity Networks [103] (SANs) and the Architecture Analysis & Design Language (AADL) [38]. In this chapter we focus on SPNs as described in Section 1.1.3.

The essence of our algorithm is that we use the description as an SPN to reduce the state space of the model into a (much smaller) graph in which each node represents a *set* of states for which the dominant paths to the goal set have the same *form*. What we mean precisely by '*paths having the same form*' will be described in the remainder of this chapter. Given the path form of the dominant path from a state to the goal set, the function d in this state is easily computed. As in Chapter 5, the algorithm described in this chapter is a pre-processing step that we run once, before we simulate, in order to determine the change of measure.

The structure of the rest of this chapter is as follows: in Section 7.1, we explain the position of this chapter in the context of the earlier work. In Section 7.2 we discuss the modelling assumptions and importance sampling scheme that are specific to this chapter. The core algorithm that determines the distance function d in an automated way is the topic of Section 7.3. Section 7.4 contains a simulation study

involving the simple model and a more realistic model. In Section 7.5, we prove that the algorithm works correctly in the sense that the calculated distance function d in a state indeed equals the shortest ϵ-distance of a path from that state to the goal set. In Section 7.6, we discuss a few challenges associated with the new method and ways to overcome them.

7.1 Context within the literature

Other efficient simulation methods that use the high-level description of a system have been proposed and implemented earlier in the literature. One way to obtain knowledge about the way the system progresses toward the goal set is, of course, to divide the transitions in the SPN into failure and repair transitions that respectively take the system towards or away from the goal set — one can then apply failure biasing as described in Section 1.3.4. This has been implemented in, among others, SAVE (see [14]) and in UltraSAN [83], the predecessor to the tool Möbius [21].

One variation of failure biasing that is especially noteworthy in the context of this chapter is distance failure biasing [18]. It is based on a notion of distance similar to the function d of (5.2). However, the technique presented in [18] can only be applied to a very narrow class of models (namely models with independent component types) and the gain compared to failure biasing may not justify the numerical effort of the minimal cut algorithm that is used (see also the discussion in [81]).

Another technique is to split the simulation effort into two different stages: one to obtain an idea about the typical behaviour to the rare set and one to use this knowledge in an importance sampling scheme. This idea forms the basis of the *cross-entropy method* for importance sampling [55, 100] and Kelling's framework for RESTART in SPN [65]. The cross entropy method has recently been implemented in the PLASMA-platform [56] for statistical model checking.

For RESTART and splitting, one implicitly divides the state space of the model in several *level sets*. Examples of how to determine these level sets are to let the user specify them by hand [78, 110], or to use a two-step approach similar to the one underlying the cross-entropy method [65]. The splitting framework has been implemented in the Stochastic Petri Net Package [110] and the tool TimeNet [118]. The methods based on this principle are largely heuristic in nature.

7.2 Model Setting

The structure of this section is as follows. In Section 7.2.1 we discuss how the notion of ϵ-distances appears in the setting of SPN-models. In Section 7.2.2, we introduce an example of an SPN that we use throughout this chapter. In Section 7.2.3, we discuss the performance property of interest, and we discuss simulation in Section 7.2.4.

7.2.1 Reliability Modelling Using Petri Nets

As described in the introduction, the model class that we consider in this chapter is that of the SPN as discussed in Section 1.1.3. In particular, states will be called *markings* and are written as vectors, i.e., $\vec{x}$ for a state instead of x, where $\vec{x} = (x_1, \ldots, x_{|P|})$, with $|P|$ the number of places. As in Chapter 5, the transition probabilities are functions of ϵ. Specifically, let $\lambda_i(\vec{x})$, $\vec{x} \in \mathcal{X}$, be the exponential rate of transition t_i, then, even though we allow $\lambda_i(\vec{x})$ to depend on the marking $\vec{x}$, we assume that numbers r_i exist such that for all $\vec{x} \in \mathcal{X}$, $\lambda_i(\vec{x}) = \Theta(\epsilon^{r_i})$, i.e.,

$$0 < \lim_{\epsilon \downarrow 0} \frac{\lambda_i(\vec{x})}{\epsilon^{r_i}} < \infty. \tag{7.1}$$

The *cost* of firing transition t_i in state $\vec{x}$ is given by $\kappa_i(\vec{x})$ defined by

$$\kappa_i(\vec{x}) = \begin{cases} r_i - \min\limits_{j=1,\ldots,|T|} \{\mathbf{1}_j(\vec{x}) r_j\} & \text{if transition } i \text{ is enabled in } \vec{x} \\ \infty & \text{otherwise} \end{cases}$$

where $\mathbf{1}_i$ is defined as in (1.1). The cost of a path through the state space is then the sum of the costs of firing the individual transitions. The cost of firing transition t is simply the distance added to a path to create a new path in which one state change more appears caused by firing t.

7.2.2 Running Example

The running example that we use throughout this chapter is a reliability model equivalent to a two-node $M/M/1$ tandem queue. It can be seen as a single component with an infinite number of hot spares; when a component or a spare breaks down, two repair phases have to be completed consecutively. Component and spares fail according to a Poisson process with rate $\lambda = \Theta(\epsilon^2)$. The times between first phase repairs are exponentially distributed with rate $\mu = \Theta(\epsilon)$. The times between second phase repairs are exponentially distributed with rate $\nu = \Theta(1)$. The transitions causing these three types of state changes are called t_1, t_2 and t_3 respectively. We assume that none of the rates depend on the marking, and that both queues will be empty most of the time. This system can be modelled using an SPN as depicted in Figure 7.1. The typical rare event that we are interested in is having n or more components awaiting the second phase of repair before all components have been repaired, starting from the first breakdown of the main component. This rare event is an instance of $\psi = \neg a \,\mathsf{U}\, b$, i.e., seeing a goal or b-state before seeing a taboo or a-state, where $(0, 0)$ is the only a-state and all states $(\cdot, z)$ with $z \geq n$ are b-states.

7.2.3 Problem Setting

Since we are interested in the estimation of π_ψ, the probability of reaching the goal set before the taboo set, we need a clear definition of the goal and taboo sets. From

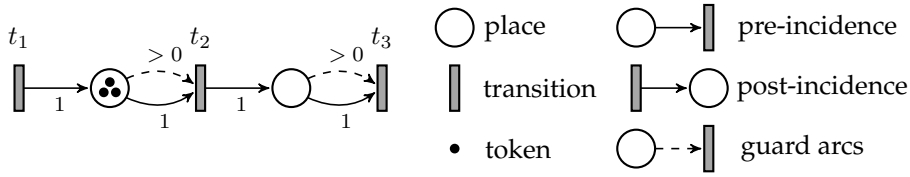


Figure 7.1: Tandem queue, depicted in the form of a stochastic Petri net. Transition t_1 fires according to rate λ, t_2 according to rate μ and t_3 according to rate ν.

now on, we will call elements of the taboo set a-markings and elements of the goal set b-markings. Since the state space is not specified explicitly, the goal and taboo sets need to be specified at the high level. The formalism that we use to achieve this is that of *guards*, the same formalism that we used to specify the firing conditions of transitions in Section 1.1.3. Formally, we let an a-guard be a 3-tuple $(p, \beta, \bowtie)$. With $G_a = \{g_a^1, \ldots, g_a^{|G_a|}\} \subset P \times \mathbb{N} \times \{\leq, \geq\}$ as the set of a-guards, we then say that a marking $\vec{x} = (x_1, \ldots, x_{|P|})$ is an a-marking if for each $g = (p_i, \beta, \bowtie) \in G_a$ it holds that $x_i \bowtie \beta$. The b-markings are then defined analogously. As mentioned in Section 1.1.5, if a marking is both an a- and b-marking, we will consider it to be a b-marking only.

Note that the definition of a-markings and b-markings given above imposes a particular structure upon the taboo and goal sets: they are only allowed to be hyper-rectangles with edges orthogonal or parallel to the axes of the state space $\mathbb{N}^{|P|}$. In other words, they must be sets of points in the interior of a Cartesian product of convex subsets of $\mathbb{R}$. An example of a set that is *not* a valid taboo or goal set in the model of Section 7.2.2 is the set of all points (x_1, x_2) such that $x_1 + x_2 \geq n$ for some $n \in \mathbb{N}$.[25] We note here that our algorithm still works when we allow the taboo set to be any union of taboo sets defined as above (similarly for the goal set). The second case study given in Section 7.4 has a non-convex goal set which is a union of hyper-rectangles as discussed above.

7.2.4 Efficient Simulation

As in earlier chapters, the change of measure used in this chapter is constructed using ZVA as discussed in Section 1.3.5, i.e., approximation of the zero variance change of measure given by (1.13). In this chapter we use the approximation $w = \epsilon^d$, giving rise to the following measure $\mathbb{Q}$:

$$q_{\vec{x}}(i) = \mathbb{Q}(\vec{x} \to \vec{x} + \vec{u}_i) = \frac{p_{\vec{x}}(i)\epsilon^{d(\vec{x}+\vec{u}_i)}}{\sum_{j=1}^{|T|} p_{\vec{x}}(j)\epsilon^{d(\vec{x}+\vec{u}_j)}}, \tag{7.2}$$

where $p_{\vec{x}}(i)$ is defined as in Section 1.1.3. The change of measure of (7.2) boils down to *'scaling out'* the ϵ-dependence of the transitions in the dominant paths.

[25] It may be possible to get around this restriction if a third place p_3 is added to the net such that each time t_1 is fired a token is placed in p_3 and a token is removed each time t_3 is fired.

Under some assumptions such as the absence of high-probability cycles and Λ — the set of states reachable from x_0 with cost $d(x_0)$— being finite this produces an estimator with the BRE-property (see Section 5.3) by Theorem 5.1. Hence, the only remaining challenge is to find $d(\vec{x})$ for each possible marking $\vec{x}$. This will be the topic of Section 7.3.

7.3 Determining the Distance Function

In this section we discuss an algorithm for finding the function d as defined in (5.2). Our aim will be to partition $\mathcal{X}$ into *zones* such that for each zone, all the states in this zone have a *similar* cost function d in a sense to be detailed below. Formally, we define a zone z to be a set of *constraints* $\{c_1^z, \ldots, c_{|z|}^z\}$ where constraints are as defined in Section 1.1.3. Let the *zone area* $\mathcal{X}_z$ be the set of states that satisfy all constraints in z.[26] The idea is then to find a set of zones Z such that the zones areas $\mathcal{X}_z, z \in Z$, form a partition of $\mathcal{X}$ and that we can find functions $d_z(\vec{x})$ that give an easy expression for the distance to $\mathcal{X}_b$ of all states $\vec{x} \in z$.

Particularly, we aim to construct a *zone graph*; a graph where the nodes correspond to the zones of Z and for which the additional condition holds that there is an arc from zones z to z' if and only if for each state $\vec{x} \in \mathcal{X}_z$ we can reach some state in z' through repeated firing of *a single transition*. We will call such a repeated firing a *stutter step*, as in, e.g., [6].

Furthermore, we want the shortest path from any state in a zone z to $\mathcal{X}_b$ to correspond to the same path through the zone graph. Finally, we want the cost in terms of ϵ-orders of firing the transition of the stutter step to be the same in all states in the same zone set. If all these conditions hold, then for each zone z we can find a function d_z that is the *same* linear function for all $\vec{x} \in \mathcal{X}_z$ (a function f is linear if $f(\vec{x}) = \vec{\alpha}^{\mathrm{T}}\vec{x} + \beta$ for some $\vec{\alpha} \in \mathbb{R}^{|P|}$ and $\beta \in \mathbb{R}$). In this section, we will clarify how this can be done.

To make the preceding concrete, consider the ***running example*** *of Section 7.2.2. The first two zones that we create are $\mathcal{X}_a$ and $\mathcal{X}_b$; in particular, $\mathcal{X}_b$ consists of the states in which $x_2 \geq n$; we assume $n \geq 3$. In the state $(1, n-1)$, t_2 needs to fire once to reach $\mathcal{X}_b$, and the cost of this step is 1. The reason is that the rate at which t_2 fires is one ϵ-order higher than that of t_3. In $(2, n-2)$, we need to fire t_2 twice, giving a total cost of 2. The same holds for all states $(y, n-y)$, $y \geq 1$; we fire t_2 y times and the total cost is y. It then makes sense to group all these states together in a zone. However, for $(n, 0)$, we need to fire t_2 n times, but the total cost is $n-1$ as t_2 does not need to compete against t_3 in the first step. Hence, $(n, 0)$ and $(n-1, 1)$ will not be in the same zone. The complete set of zones resulting from the algorithm developed in this section, where for each zone we display its cost function and shortest path to the taboo set, is illustrated in Figure 7.2.*

Each path through the zone graph corresponds to a *path form* in the state space (the concept of path forms will play a particularly important role in Section 7.5). In

[26] We sometimes abuse this terminology by saying that a state x is inside a zone z; by this we mean that x is in the zone area $\mathcal{X}_z$.

Section 7.3.1, we will outline the main loop of the algorithm. An initial partitioning is always necessary, as we will discuss in Section 7.3.2. However, this initialisation alone is not sufficient. It may be that it is not possible for all markings in an initial zone to reach another zone by the same stutter step; this is the topic of Section 7.3.3. Also, there may exist markings within a single zone for which the shortest path follows a different sequence of stutter steps, as we will discuss in Section 7.3.4. In Section 7.3.5 we discuss how we update some global variables after each call to the main loop.

7.3.1 Main Loop

Let a *stutter step* s be a triple (z^o, t_j, z^d), where z^o is the source/origin zone, z^d is the destination zone and t_j is the transition that is repeatedly fired (we note here that a list of global variables such as z^o and z^d is given in Table 7.1). The algorithm works as follows: we keep a list S of stutter steps that could be part of shortest paths. After initialising the list, we repeatedly take stutter steps s out of S and check whether for all markings in the origin zone of s it holds that

1) we can indeed reach the destination zone of s using only the given stutter step, and

2) the new distance function indeed gives shorter distance than what was known before.

If not, we split the source zone and (potentially) add new stutter steps to S. Finally, we discard s, pick a new stutter step, and repeat until S is empty. The precise way in which this is done is given by Algorithm 7.1.

Algorithm 7.1 Main loop.

```
1: initZoneGraph()
2: while S ≠ ∅ do
3:     s = (z^o, t_j, z^d) := some element from S
4:     possibilitySplit(s)
5:     if d_{z^o} ≠ unassigned then costSplit(s)
6:     update();
7: end while
```

7.3.2 Initialisation Phase (`initZoneGraph`)

During the initialisation phase, the state space is divided into zones such that

(a) from all states in the same zone set the same transitions are enabled, and

(b) the states in a zone set are either all in $\mathcal{X}_a$, all in $\mathcal{X}_b$, all in both, or all in neither.

Global Variables	
Z	the set of zones
V	the set of (stutter) steps connecting the zones
S	set of steps to consider
s	step currently under investigation
y	number of times t_i is fired at zone switch
ρ	boolean indicating whether s was relevant
Z^o	set of new zones due to s
d_z	distance function in zone z
d_z	distance function in zone z
z^o	source node of a step s
z^d	destination node of a step s
z^n	zone in Z^o that can reach z_d by firing t_i and for which d is lower than known previously
θ_z	the path form corresponding to the distance function in zone z (see Section 7.5)
$\mathcal{D}$	The set of path forms considered so far (see Section 7.5).

Table 7.1: List of global variables used in the algorithm.

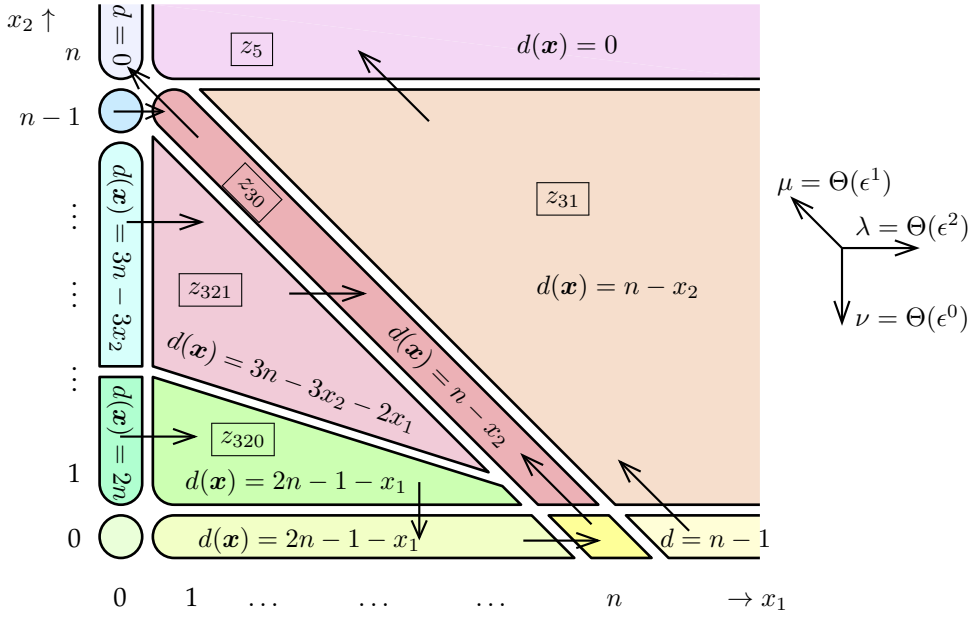


Figure 7.2: The final result of a call to the algorithm.

Condition (a) implies that the cost of firing a transition is always the same in a zone (because the cost depends on which other transitions can be fired). Hence, instead of the $\kappa_j(\vec{x})$ of (7.1) we can write $\kappa_j(z)$ where $\kappa_j(z) = \kappa_j(\vec{x})$ for all $\vec{x} \in \mathcal{X}_z$. During the initialisation we can already assign distance ∞ to the states in $\mathcal{X}_a$ and 0 to the states in $\mathcal{X}_b$. Furthermore, we initialise the stutter step list S during this phase; its initial elements will be those stutter steps that directly lead into $\mathcal{X}_b$. The precise way in which all this is done is given in Algorithm 7.2.

In line 2 of Algorithm 7.2, we use the negation $\neg c$ of a constraint c. If $c = (\alpha, \beta, \leq)$, then its negation is given by $\neg c = (\alpha, \beta + 1, \geq)$, and if $c = (\alpha, \beta, \geq)$ then $\neg c = (\alpha, \beta - 1, \leq)$. If all elements of α are at least 1, then the resulting zone sets $\mathcal{X}_{\{c\}}$ and $\mathcal{X}_{\{\neg c\}}$ are each other's complements with respect to $\mathcal{X}$.

Lines 9 and 10 are not necessary for the proper functioning of the algorithm but aid understanding of the proof of correctness. They are discussed further in Section 7.5.

Algorithm 7.2 initZoneGraph()

1: $C := \{c = (p, \beta, \bowtie) : (p, \cdot, \beta, \bowtie) \in G \vee c \in G_a \cup G_b\}$
2: $Z := \{z \in \mathcal{Z} : \forall c \in C : c \in z \vee \neg c \in z, \mathcal{X}_z \neq \emptyset\}$ ▷ $\mathcal{Z}$ = all possible zones
3: $V := \{(z', t_i, z'') : z', z'' \in Z, z' \neq z'', \exists \vec{x} \in \mathcal{X}_{z'}, \vec{x} + \vec{u}_i \in \mathcal{X}_{z''}, t_i \in T\}$
4: $Z_a := \{z \in Z : \forall \vec{x} \in \mathcal{X}_z : \vec{x} \in \mathcal{X}_a\}$
5: $\forall z \in Z_a : d_z := \infty$
6: $\forall z \in Z_a : \theta_z = \emptyset$
7: $Z_b := \{z \in Z : \forall \vec{x} \in \mathcal{X}_z : \vec{x} \in \mathcal{X}_b\}$
8: $\forall z \in Z_b : d_z = 0$
9: $\forall z \in Z_b : \theta_z = (z)$ ▷ see Section 7.5
10: $\mathcal{D} := \{\theta : \theta = (z), z \in Z_b\}$ ▷ see Section 7.5
11: $S := \{v \in V : v = (z, \cdot, z'), z \notin Z_a, z' \in Z_b\}$

*For the **running example** as displayed in Figure 7.1, the transition structure first gives us four initial zones: z_0 where only t_1 can fire, z_1 for t_1 and t_2, z_2 for t_1 and t_3, and z_3 for all three. The zone structure resulting from a call to initZoneGraph() is displayed in Figure 7.3(a). We obtain two additional zones, z_4 and z_5, to distinguish $\mathcal{X}_b$. S is initialised with all stutter steps leading into these two zones; the only stutter steps satisfying this requirement are the two t_2-stutter steps going from z_3 into z_4 and z_5.*

7.3.3 Divide Zones by Path Existence (`possibilitySplit`)

To determine the cost of a stutter step $s = (z^o, t_j, z^d)$, we need to determine the number of times y that t_j must fire to take a marking in z^o to z^d. This is done by `findNumberOfTransitions`. The main idea is to find a function $y(\vec{x})$ (written as y for brevity) such that after firing t_j $y - 1$ times, the marking is still in z^o, and after firing one more time the marking is in z^d. In order to find this number, we choose any constraint c_1 from z^o and c_2 from z^d that exclude each other, i.e., $\mathcal{X}_{z^o} \cap \mathcal{X}_{z^d} = \emptyset$, and choose y to be the smallest number of firings to enable c_2. Since all constraints

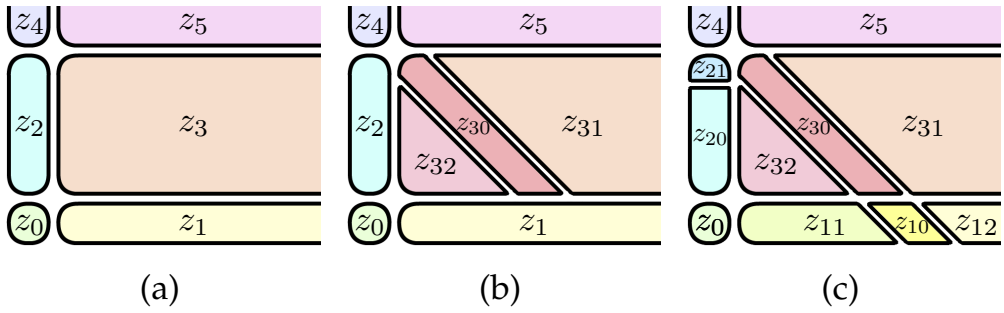


Figure 7.3: Figures (a-c) depict the zones after several iterations of the algorithm.

are non-strict inequalities, y is chosen such that $\vec{x} + y\vec{u}_j$ exactly satisfies the constraint. The remaining constraints in z^o and z^d then impose restrictions on $\vec{x}$ that must be satisfied in order for this stutter step to be possible.

Algorithm 7.3 possibilitySplit().

Require: stutter step s

1: $(c_1, c_2) :=$ some two constraints such that
   1) $c_1 \in z^o$, 2) $c_2 \in z^d$ and 3) $\mathcal{X}_{\{c_1\}} \cap \mathcal{X}_{\{c_2\}} = \emptyset$
2: $y :=$ findNumberOfTransitions(c_2,$\vec{u}_i$)
3: $C_1 := \{c : (c = [\vec{a}(\vec{x} + (y-1)\vec{u}_i) \bowtie b]) \wedge ([\vec{a}\vec{x} \bowtie b] \in z^o \backslash c_1)\}$
4: $C_2 := \{c : (c = [\vec{a}(\vec{x} + y\vec{u}_i) \bowtie b]) \wedge ([\vec{a}\vec{x} \bowtie b] \in z^d \backslash c_2)\}$
5: $C := C_1 \cup C_2$
6: $Z^o := \{z : \forall c \in C : c \in z \vee \neg c \in z \wedge \forall c \in z^o : c \in z \wedge \exists \vec{x} \in \mathcal{X} : \vec{x} \in \mathcal{X}_z\}$
7: $z^n := z \in Z^o : \forall c \in C : c \in z$
8: $d_{z^n}(\vec{x}) := d_{z^d}(\vec{x} + y\vec{u}_i) + y\kappa_{z^o}(j)$

Assume that we happen to first consider the μ-stutter step from z_3 to z_4. After the initialisation phase, there are two pairs of constraints from z_3 and z_4 that exclude each other; the pair $x_1 \geq 1$ and $x_1 \leq 0$, and the pair $x_2 \leq n-1$ and $x_2 \geq n$. If we consider the first pair, we end up with $y = x_1$. The two constraints that we end up through lines 4 and 5 of Algorithm 7.3 are $x_1 + x_2 - 1 \leq n-1$ and $x_1 + x_2 \geq n$. If we would consider the second pair, we would have found $y = n - x_2$, leading to the same restrictions on $x_1 + x_2$.

In line 5, we construct the set C of constraints that must be satisfied for the stutter step s to be taken. We do this by letting C be the union of C_1, the set of constraints that impose that after $y-1$ firings of t_i we are in z^o, and C_2, the set of constraints that impose that after y firings of t_i we are in z^d. Given C, the zone z^o may need to be subdivided such that one zone remains in which the stutter step s is always possible. This is done in line 6 of `possibilitySplit`; all zones that consist of combinations of constraints in C or their negations are considered. If such a zone is non-empty (which is checked using an Integer Linear Programming

(ILP)-solver, although this can be computationally expensive), it is added to Z^o, the set of new zones. The zone z^n is the subzone (i.e., a subset in terms of constraints) of z^o for which s was possible.

Since we obtained the additional constraints $x_1 + x_2 \leq n$ and $x_1 + x_2 \geq n$ for the running example, we obtain three new non-empty zones; z_{30}, z_{31} and z_{32}, all depicted in Figure 7.3(b). Of those, z_{30} has cost $d_{z_{30}}(\vec{x}) = x_1$ or, equivalently, $d_{z_{30}}(\vec{x}) = n - x_2$, depending on which of the two constraint pairs was considered. The other two zones do not have any cost assigned yet. When the function `update` *in Algorithm 7.1 is called, the stutter steps from z_1, z_2, z_{31} and z_{32} to z_{30} are added to S. Furthermore, the stutter step from z_3 to z_5 is removed, as z_3 no longer exists. It is replaced by the μ-stutter step from z_{31} to z_5.*

Upon further calls to `possibilitySplit`*, the zone z_1 is subdivided into three new zones and z_2 into two new zones, and distance functions are assigned to all. This is displayed in Figure 7.3(c). Furthermore, zones z_{31} and z_{32} have distances assigned to them. In particular, we mention the distance function of z_{32}: $d_{z_{32}}(\vec{x}) = 3n - 2x_1 - 3x_2$. In the next section, z_{32} is split into two zones, only one of which retains this distance function.*

7.3.4 Divide Zones by Costs (`costSplit`)

When the algorithm as described so far is executed, it will consecutively consider zones to which no distance function has yet been assigned, split them and assign costs to them. However, when a zone is considered that *already has* a distance function assigned to it, the new path may be the shortest only for a subset of the zone. We need `costSplit` for these situations.

Algorithm 7.4 costSplit()

Require: step s

1: $c_n := d_{z^n}(\vec{x}) - d_{z^o}(\vec{x}) < 0$
2: $z'' := z^n \cup \{\neg c_n\}$
3: $z^n := z^n \cup \{c_n\}$
4: $d_{z''}(\vec{x}) := d_{z^o}(\vec{x})$
5: **if** $\exists \vec{x} \in \mathcal{X} : \vec{x} \in \mathcal{X}_{z^n}$ **then**
6: **if** $\nexists \vec{x} \in \mathcal{X} : \vec{x} \in \mathcal{X}_{z''}$ **then**
7: $z^n := z^n \backslash \{c_n\}$ ▷ After all, we apparently do not need c_n.
8: **else**
9: $Z^o := Z^o \cup \{z''\}$
10: **end if**
11: **end if**

Say that, after running Algorithm 7.3, one has found a subzone z^n of z^o for which the stutter step under consideration can be applied, and for which d_{z^n} is the distance function. If d_{z^o} has already been assigned, then the stutter step under consideration is only interesting for those markings $\vec{x}$ for which $d_{z^n}(\vec{x}) < d_{z^o}(\vec{x})$. This constraint is exactly the one constructed in line 1 of Algorithm 7.4. The zone z^n is then divided into two new zones: the part where the constraint holds remains

z^n, and z'', for which it does not. If z^n is empty, the stutter step under consideration has been irrelevant, and the list S should not be updated. If only z'' is empty, then z^n fully replaces z^n. However, if both z^n and z'' are non-empty, the two of them are added to Z^o instead of just z^n.

*For the **running example**, the distance function $d_{z_{32}}(\vec{x}) = 3n - 2x_0 - 3x_1$ had already been assigned to the zone z_{32} as depicted in Figure 7.3(c). Assume that the next stutter step to be considered is the t_3-stutter step from z_{32} to z_{11}. Since the distance function in z_{11} is $2n - 1 - x_1$, and the cost of firing t_3 in z_{32} is zero, the new zones z_{320} and z_{321} are separated by the line $3x_2 \leq n - x_1$. In the next and final iteration z_{21} is further divided into the zones z_{210} and z_{211} by* `possibilitySplit`.

Algorithm 7.5 update()

$\theta^* := (z^o, t_j, z_0^\theta, t_0^\theta, \ldots, z_{|\theta|}^\theta, t_{|\theta|}^\theta) : \theta = \theta_{z^d}$ ▷ so θ^* is the path form of s
$\mathcal{D} := \mathcal{D} \cup \{\theta^*\}$ ▷ followed by θ_{z^d}, see Section 7.5
if $\mathcal{X}_{z^n} \neq \emptyset$ **then**
  $\theta_{z^n} = \theta^*$ ▷ see Section 7.5
  alterS(Z^o,z^o)
  $S := S \cup \{(z', \cdot, z^n) \in V : z' \neq z^n\}$
  $C := \emptyset$
  for all $s \in V : s = (z^n, t_i, z^n)$ **do**
    if $d_{z^n}(0) - d_{z^n}(\vec{u}_i) > \kappa_i(z^n)$ **then**
      $C := C \cup \{c : c = \neg(\vec{a}(\vec{x} + \vec{u}_i) \bowtie b) \ \wedge \ \vec{a}\vec{x} \bowtie b \in z^n\}$
    end if
  end for
  $Z^n := \{z : (\forall c \in C : c \in z \vee \neg c \in z) \wedge (\forall c \in z^n : c \in z) \wedge (\exists \vec{x} \in \mathcal{X} : \vec{x} \in \mathcal{X}_z)\}$
  $\forall z \in Z^n : d_z := d_{z^n}, \theta_z := \theta^*$
  alterS(Z^n,z^n)
  $S := S \cup \{s' \in V : s' = (z', \cdot, z''), z' \neq z'', z' \in Z^n, z'' \in Z^n\}$
end if

Algorithm 7.6 alterS(Z^*,z^*)

Require: set Z^*, zone z^*
if cardinality of $Z^* > 1$ **then** ▷ I.e., if new zones have been created
  $Z := (Z \backslash \{z^*\}) \cup Z^*$
  $V := \left\{(z', t_j, z'') : \begin{array}{l} \exists \vec{x} : \vec{x} \in \mathcal{X}_{z'}, \vec{x} + \vec{u}_j \in \mathcal{X}_{z''} \\ z', z'' \in Z \end{array}\right\}$
  $S := S \cup \{(z'', t, z') \in V : z'' \in Z^* \wedge (z^*, t, z') \in S\}$
  $S := S \backslash \{v \in S : v = (z^*, \cdot, \cdot)\}$
  $S := S \cup \{(z', t, z'') \in V : z'' \in Z^* \wedge (z', t, z^*) \in S \wedge (z', t, z'') \notin S\}$
  $S := S \backslash \{v \in S : v = (\cdot, \cdot, z^*)\}$
end if

7.3.5 Update the step list (`update` and `alterS`)

At the end of each iteration of the main loop, the routine `update` (i.e., Algorithm 7.5) is called in order to add new steps to S and to update existing steps in S that may need to change due to the partitioning of a zone. The routine begins with several steps that do not have an impact on the zone partitioning but which help illustrate the proof in Section 7.5: we let θ^* be the path form considered and update $\mathcal{D}$ (more detail is given in Section 7.5). We then begin the main part of the routine by checking whether $\mathcal{X}_{z^n}$ is empty — since z^n corresponds to the zone that is constructed by `possibilitySplit` and `costSplit` as the zone for which the stutter step is both possible and cost-efficient, $\mathcal{X}_{z^n}$ is the set of states for which it makes sense to carry out the considered stutter step. If this set is empty, we do not do anything. Otherwise, we let the new dominant path form of z^n be the considered path form and call the routine `alterS` on z^o and Z^o.

In `alterS` (i.e, Algorithm 7.6), we first check if the zone given as an argument has been partitioned. If so, we update the zone graph (Z, V) and reroute steps already in S that use the partitioned zone such that they use its partitions. In line 6 of `update` we add to S all the steps that lead to z^n. If some distance had previously been assigned to z^n or a bigger zone, then some of the steps added in line 6 of `alterS` may have been added previously in line 6 of `update` . This is the reason why the implementation of the algorithm does not add duplicates, as implied by the condition $(z', t, z^*) \notin S$ that appears in line 6 of `alterS`.

One restriction imposed in line 6 of `update` is that a step $s = (z^o, t_j, z^d)$ that is added to S in this line cannot stay in the same zone (i.e., $z^o \neq z^d$). The purpose of lines 7 to 16 is to handle these cases correctly. As we argue in Section 7.5, if it is optimal to fire a transition and remain within the same zone then it is optimal to fire this transition as often as possible. Specifically, it is only optimal to fire a transition within the same zone if the cost of firing the transition is less than the gain in terms of distance, i.e., if $d_{z^n}(0) - d_{z^n}(\vec{u}_i) > \kappa_i(z^n)$, which is checked in line 10. If this condition holds, we add constraints in line 10 of `update` that indicate whether it is still possible to fire the relevant transitions or not. The zone z^n is then further partitioned using these constraints, and steps already in S involving z^n are updated in `alterS`. Finally, the steps between the new zones are added to S in line 16. Note that we now do not need to consider the steps that remain within the same zone, as these steps would not fire the relevant transition as often as possible.

7.4 Empirical Results

In this section, we empirically demonstrate the effectiveness of our algorithm. In Section 7.4.1, we describe the case studies that we will consider. We present numerical results obtained using the algorithm to find d in Section 7.4.2, while in Section 7.4.3 we use d to apply simulation.

7.4.1 Case Description

We use two case studies. The first is the running example from Section 7.2.2, where the system is assumed to have failed if $x_2 > n$, $n \in \mathbb{N}$.

The second is a more realistic multicomponent system with interdependent component types, taken from [96]. For the latter we have six component types, with n_i components of type i and $(n_1, \ldots, n_6) = (n+2, n+1, n+3, n, n+4, n+2)$. In the benchmark setting, $n = 3$. If k components of type i have failed, the rate at which the next component of type i fails is $(n_i - k)\lambda_i\epsilon$, where $(\lambda_1, \ldots, \lambda_6) = (2.5, 1, 5, 3, 1, 5)$. There is a single repairman who repairs components following a preemptive priority repair strategy, where components of type i have priority over components of type j if $i < j$. The repair rate for type i is always μ_i, $(\mu_1, \ldots, \mu_6) = (1, 1.5, 1, 2, 1, 1.5)$. The system is said to have failed when all components of any type are down. We estimate the probability that, after the first component failure (drawn randomly), the system fails before all components are repaired.

7.4.2 Results of the Distance Finding Algorithm

A summary of the results of our algorithm is displayed in Table 7.2. The number of initial constraints is the main factor that determines the runtime of the algorithm. For the initial zones, we distinguish between the $(a \cup b)$- and $\neg(a \cup b)$-zones because only the latter have an impact on the runtime of the rest of the algorithm. A few things to mention: the number of zones may depend on n because for small n some zones will be empty, which are discarded. Also, the final number of zones may depend on the way stutter steps are chosen from S in the main loop, because if a zone is split by a stutter step that later turns out to be insignificant, these zones are not recombined by our implementation, so both the number of zones and the number of iterations are implementation-dependent. For the multicomponent system, one observes that the number of both initial and final zones is much lower in Table 7.2 than in the comparable table in [93]. The reason in that in [93] the need for lines 7 to 16 of `update` was averted by including the creation of '*margins*' around the initial zones in `initZoneGraph`. Obviously, the implementation that includes lines 7 to 16 gives much better performance. This also has a slight impact on the running example, but the consequences in this setting are nowhere near as profound as for the multicomponent system.

7.4.3 Simulation Results

The simulation results are summarised in Tables 7.3 and 7.4. In both tables, we display the results for three simulation methods: standard Monte Carlo (MC), Balanced Failure Biasing (BFB) and IS based on our distance finding algorithm (Zone-IS). As discussed in Section 1.3.4, BFB means that the total probability of firing a failure transition is set to $\frac{1}{2}$, uniformly distributed over the individual failure transitions (similarly for the repairs). In our implementation of BFB, we only consider the

	Running Example		Multicomponent System	
n	3	10	3	5
# initial constraints	5	5	18	18
# initial zones	6	6	729	729
# initial $\neg(a \cup b)$-zones	3	3	63	63
# final $\neg(a \cup b)$-zones	8	10	1444	1885
# iterations in main loop	28	39	12879	27378
# markings in $\mathcal{X}$	∞	∞	40320	241920
time to construct (sec)	0.45	0.5	46.03	173.96

Table 7.2: Results of the numerical analysis for the running example.

ν-transition t_3 to be a repair transition. Next to the simulation results, we display numerical approximations obtained using the model checking tool PRISM [67].

For the efficiency of the methods we look at the width of the 95%-confidence intervals (CI) of the estimates. A lower value generally means a better estimate; however, if a change of measure is poorly suited for the system, IS may suffer from underestimation [33]. An example of this are the results for BFB for $n = 10$ and $\epsilon = 0.01$ in Table 7.3. For the sake of consistency with [96], we used $200\,000\,000$ runs per MC-estimate and $1\,000\,000$ runs per IS-estimate. In all cases Zone-IS outperforms BFB, except for $n = 5$ in Table 7.4. The reason is that BFB needs a clear distinction between failures and repairs to work well.

n	ϵ		$\hat{\pi}$	95%-CI-bounds	N	π (PRISM)
3	10^{-1}	MC	$1.09{\cdot}10^{-4}$	$\pm\, 0.01400{\cdot}10^{-4}$	200 000 000	$1.100{\cdot}10^{-4}$
		BFB	$1.088{\cdot}10^{-4}$	$\pm\, 0.015{\cdot}10^{-4}$	1 000 000	
		Zone-IS	$1.099{\cdot}10^{-4}$	$\pm\, 0.001{\cdot}10^{-4}$	1 000 000	
	10^{-2}	MC	$5.0{\cdot}10^{-9}$	$\pm\, 9.8{\cdot}10^{-9}$	200 000 000	$1.010{\cdot}10^{-8}$
		BFB	$9.959{\cdot}10^{-9}$	$\pm\, 0.209{\cdot}10^{-9}$	1 000 000	
		Zone-IS	$1.01016{\cdot}10^{-8}$	$\pm\, 0.00004{\cdot}10^{-8}$	1 000 000	
5	10^{-1}	MC	$1.0{\cdot}10^{-8}$	$\pm\, 1.386{\cdot}10^{-8}$	200 000 000	$1.100{\cdot}10^{-8}$
		BFB	$1.099{\cdot}10^{-8}$	$\pm\, 0.084{\cdot}10^{-8}$	1 000 000	
		Zone-IS	$1.100{\cdot}10^{-8}$	$\pm\, 0.003{\cdot}10^{-8}$	1 000 000	
	10^{-2}	MC	—	—	200 000 000	$1.010{\cdot}10^{-16}$
		BFB	$9.036{\cdot}10^{-17}$	$\pm\, 1.529{\cdot}10^{-17}$	1 000 000	
		Zone-IS	$1.0100{\cdot}10^{-16}$	$\pm\, 0.0002{\cdot}10^{-16}$	1 000 000	
10	10^{-1}	MC	—	—	200 000 000	$1.100{\cdot}10^{-18}$
		BFB	$1.018{\cdot}10^{-18}$	$\pm\, 1.715{\cdot}10^{-18}$	1 000 000	
		Zone-IS	$1.111{\cdot}10^{-18}$	$\pm\, 0.021{\cdot}10^{-18}$	1 000 000	
	10^{-2}	MC	—	—	200 000 000	$1.010{\cdot}10^{-36}$
		BFB	$8.791{\cdot}10^{-38}$	$\pm\, 17.231{\cdot}10^{-38}$	1 000 000	
		Zone-IS	$1.007{\cdot}10^{-36}$	$\pm\, 0.002{\cdot}10^{-36}$	1 000 000	

Table 7.3: Results of the simulation analysis for the running example.

n	ϵ		$\hat{\pi}$	95%-CI-bounds	N
3	10^{-3}	MC	$7.15{\cdot}10^{-7}$	$\pm\, 1.17{\cdot}10^{-7}$	200 000 000
		BFB	$7.406{\cdot}10^{-7}$	$\pm\, 0.285{\cdot}10^{-7}$	1 000 000
		Zone-IS	$7.282{\cdot}10^{-7}$	$\pm\, 0.131{\cdot}10^{-7}$	1 000 000
	10^{-4}	MC	—	—	200 000 000
		BFB	$4.860{\cdot}10^{-9}$	$\pm\, 0.265{\cdot}10^{-9}$	1 000 000
		Zone-IS	$4.878{\cdot}10^{-9}$	$\pm\, 0.016{\cdot}10^{-9}$	1 000 000
5	10^{-3}	MC	—	—	200 000 000
		BFB	$1.329{\cdot}10^{-10}$	$\pm\, 0.335{\cdot}10^{-10}$	1 000 000
		Zone-IS	$8.880{\cdot}10^{-11}$	$\pm\, 0.953{\cdot}10^{-11}$	1 000 000
	10^{-4}	MC	—	—	200 000 000
		BFB	$1.532{\cdot}10^{-15}$	$\pm\, 0.790{\cdot}10^{-15}$	1 000 000
		Zone-IS	$1.994{\cdot}10^{-15}$	$\pm\, 0.044{\cdot}10^{-15}$	1 000 000

Table 7.4: Results of the simulation analysis for the multicomponent system.

7.5 Proof of Correctness

In this section, we prove that when the algorithm described in this chapter terminates, for each state $\vec{x} \in \mathcal{X}$ it holds that $d(\vec{x})$ satisfies the definition given in (5.2) — i.e., $d(\vec{x})$ represents the smallest ϵ-distance from $\vec{x}$ to the goal set. Before we continue with the proofs, we give more detail about the operation of our algorithm. The main goal of the algorithm is to group states together such that for each state $\vec{x}$ in the same zone the most likely path from $\vec{x}$ to the goal set has the same *path form*. As said in the introduction to Section 7.3, path forms are a way of characterising groups of paths through the zone graph, where the zone graph is the graph consisting of the relevant zones in the system and the connecting stutter steps. A complication is that the zone graph is constantly updated; while we run the algorithm zones may be subdivided into other zones. Hence, we need a notion of path forms that is more consistent throughout the running of the algorithm.

Let $\mathcal{Z}$ be the set of all zones, i.e., the set of all sets of constraints (note that this set is much larger than the set of all zones in the zone graph during a specific iteration of the algorithm). Formally, let a path form θ be given by $(z_0^\theta, t_0^\theta, z_1^\theta, \ldots, t_{|\theta|-1}^\theta, z_{|\theta|}^\theta)$ where $\forall i : z_i^\theta \in \mathcal{Z}$ and $t_i^\theta \in T$. A path ω given by $(x_0^\omega, x_1^\omega, \ldots, x_{|\omega|}^\omega)$ which is the result of firing the sequence of transitions $(t_1^\omega, \ldots, t_{|\omega|}^\omega)$ is said to *correspond to* a path form θ iff $x_0^\omega \in \mathcal{X}_{z_0^\theta}$ and

$$\exists n_0, \ldots, n_{|\theta|} \text{ s.t. } \begin{cases} \forall i, j = 1, \ldots, |\theta| - 1 : n_i \in \{1, \ldots, |\omega| - 1\} \text{ and } n_i < n_j, \\ \forall i = 1, \ldots, |\theta| : \forall n = n_{i-1}, \ldots, n_i - 1 : x_n^\omega \in \mathcal{X}_{z_{i-1}^\theta} \text{ and } t_n^\omega = t_{i-1}^\theta, \\ n_0 = 0,\ n_{|\theta|} = |\omega| \text{ and } x_{|\omega|}^\omega \in \mathcal{X}_{z_{|\theta|}^\theta}. \end{cases}$$

The most trivial path forms are those with $|\theta| = 0$; these paths consist only of a single zone and no stutter steps.

While running the algorithm, we are principally interested in assigning to each relevant zone z its distance function d_z. However, these distance functions are the same linear functions for all states in a zone because for all states in a zone the shortest paths to the goal set have the same path form. To illustrate the proof given in this section, we also keep track of the path forms to which the shortest paths correspond in the algorithm: θ_z is the path form for the dominant paths in zone z.

During initialisation, we assign to the zones that consist only of goal (i.e., b-) states (which exist because we use the constraints that define the b-states for the initial partitioning in `initZoneGraph`), the trivial path forms consisting only of these zones and no stutter steps. In subsequent iterations of the main loop path forms are added to other zones in line 4 of `update`; the added path form is the one constructed in line 1 of the same routine. These subsequent path forms are constructed iteratively by prefixing for each considered step s its source z^o and transition t_j to the path form θ_{z^d} that was previously assigned to the destination z^d of s; this is done in line 1 of `update`. Since each path form is simply an extension of a previously considered path form, the path forms leading to the goal zones can be seen as having a tree-like structure as depicted in Figure 7.4. In Figure 7.4, each node is a zone and each edge connecting the zones is labelled with a single transition. Each zone consisting only of b-states is the root of a different tree. Zone z is a child of zone z' in the tree via an edge labelled with transition t if there exists a state in z such that if we repeatedly fire the transition t we reach a state in z'. In each iteration of the algorithm one of the nodes in the tree is considered, and if a node is considered then all its ancestor nodes have been considered before it. A tree represents the same behaviour as the zone graph, albeit in a form that more closely resembles the way the algorithm operates.

To keep track of which path forms have been considered, we use the set $\mathcal{D}$ which is initialised in line 10 of `initZoneGraph`. Every time the main loop is concluded we add the considered path form to $\mathcal{D}$ in line 2 of `update`. The set $\mathcal{D}$ is updated after each iteration of the main loop, but we refer to the set that corresponds to $\mathcal{D}$ after the ith iteration as $\mathcal{D}_i$. We note here that path forms may be added to $\mathcal{D}$ that contain zones that are partitioned in later stages of the algorithm, i.e., $\mathcal{D}$ may contain paths that contain zones that are no longer in the zone graph. This is not a problem; if a dominant path exist of a form that contains a zone z that is later partitioned, then in the zone graph with z having been partitioned there must be a path form corresponding to this dominant path. To see this, note that when a zone z is partitioned we add all possible stutter steps between the new zones to the zone graph in line 3 of `alterS`. Since the new zones form a partition of z, each path going through z consists of a series of states and transitions that must run through the new zones, and since the stutter steps leading this path from one new zone to another must be in the zone graph there must a full path form in the new zone graph corresponding to the dominant path.

The first important property underpinning the algorithm is the invariant of Proposition 1 which says that during each iteration of the algorithm the function d assigned to each state in $\mathcal{X}$ is the lowest among all distances corresponding to

path forms added to $\mathcal{D}$ so far. Having proved this invariant, we then know that if we search each node in each tree (remember that we have a tree for each zone consisting of b-markings), we will have found the optimal solution for each state $\mathcal{X}$. Unfortunately, in realistic models the number of nodes in the trees is typically infinite. As a remedy, the algorithm has several built-in conditions that say that for certain nodes we can ignore the entire subtree rooted in the node. This will be the subject of Proposition 2: proving that the nodes in these subtrees indeed cannot correspond to dominant paths. With Proposition 1 and 2 combined, we know that when the algorithm terminates the functions d returned indeed correspond to the dominant paths.

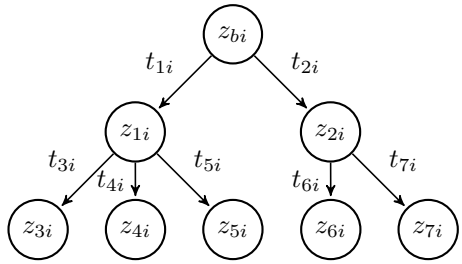


Figure 7.4: Representation as a tree of the path forms leading to a zone z_{bi} of b-markings.

Proposition 1. *The algorithm satisfies the following invariant: if, after step i of the algorithm, cost $d_i(\vec{x})$ has been assigned to state $\vec{x} \in \mathcal{X}$, then*

1) there exists a path from $\vec{x}$ to some state in $\mathcal{X}_b$ with cost $d_i(\vec{x})$, and

2) there does not exist a path from $\vec{x}$ to some state in $\mathcal{X}_b$ of a form in $\mathcal{D}_i$ that has cost lower than $d_i(\vec{x})$.

Proof. This is proved using induction on i.

Initialisation ($i = 0$):

For all b-states we have cost 0 and for all others cost ∞. This means that *1)* holds because the empty paths (which consist of no stutter steps) give these results. Furthermore, *2)* holds because no other path forms have been considered than the empty paths.

Induction

Assume that the invariant holds after step $i-1$. At step i, we consider a unique path form ω_i, given by

$$z^o \xrightarrow{t_j} z^d \longrightarrow \ldots \longrightarrow z^{\omega_i}_{|\omega_i|}.$$

Consider the first stutter step. In a path with path form ω_i, this means that we are in a state $\vec{x} \in \mathcal{X}_{z^o}$ and through firing t_j a total number of $y(\vec{x})$ times end up in state $\vec{x}' \in \mathcal{X}_{z^d}$. From $\vec{x}'$, the rest of the path is so far optimal by the induction hypothesis.

1) If $d_{i-1}(\vec{x}) = d_i(\vec{x})$, *1)* is evident. If $d_{i-1}(\vec{x}) \neq d_i(\vec{x})$, then by Algorithm 7.3 we know that the stutter step must be possible (or else $\vec{x}$ would not satisfy the requirement imposed on z^n in Line 7), so *1)* also holds here.

2) First, if $d_i(\vec{x}) \neq d_{i-1}(\vec{x})$ it must hold that that $d_i(\vec{x}) < d_{i-1}(\vec{x})$, because of the constraint imposed on z^n in Line 1 of Algorithm 7.4. So $d_i(\vec{x}) \leq d_{i-1}(\vec{x})$, which (in combination with the induction hypothesis) means that *2)* is only false if there exists a path of the form ω_i that gives cost lower than $d_i(\vec{x})$. This would mean that there exists a function $y'(\vec{x}) \neq y(\vec{x})$ that results in a path with lower cost. Firing it less than y times cannot lead from $\vec{x}$ to $\mathcal{X}_{z^d}$ due to the definition of $y(\vec{x})$ as the smallest number of times needed to cross from z^o into z^d. Firing t more than $y(\vec{x})$ times would violate the definition of the path form as this mean that the transition is also fired in a different zone. Also, we can assume that $z^o \neq z^d$ as this is a condition imposed in both line 6 and 16 of `update`. Hence, *2)* always holds. □

Proposition 2. *If a path is dominant then it corresponds to a path form that is in the final version of $\mathcal{D}$, or there exists a path with a path form in the final version of $\mathcal{D}$ with the same cost.*

Proof. We first consider which path forms are never added to $\mathcal{D}$. In each step of the algorithm we consider a step (z^o, t_j, z^d) from S where we call z^o the source zone and z^d the destination zone. For a path *not* to be in $\mathcal{D}$ one of its steps must not have been considered, and for that step not to have been considered it mush never have been added to S. Hence, we will focus on which steps are never added to S.

Steps are added to S in line 6 and in line 16 of `update`, and in `alterS`. The purpose of `alterS` is to alter steps in S that involve a zone that is to be partitioned such that these steps use the new zones. Accordingly, one easily checks that for a step to be added to S in `alterS`, a step must already exist in S that has the partitioned zone as a source or destination. Hence, we only consider the steps that are added to S in line 6 or in line 16 of `update`.

First, we consider those steps added to S in line 6. Assume that $s = (z^o, t_j, z^d)$ has just been considered, this means that $\mathcal{X}_{z^o}$ has been partitioned into $(\mathcal{X}_z)_{z \in Z^o}$ where it may be that $Z^o = \{z^o\}$. The most interesting element of Z^o is the zone z^n. All those $s' = (z', t_k, z'')$ are added to S for which $z'' = z^n$. Steps with/to $z'' \in Z^o \backslash \{z^n\}$ are *never* added to S. We know that z^n equals z^o with the additional constraints imposed in lines 3 and 4 of `possibilitySplit` and line 1 of `costSplit`. Furthermore, we do not add (z', t_k, z'') to S if $z' = z''$ (this case is handled in lines 7 to 16) or if there does not exist a path that corresponds to (z', t_k, z'') because otherwise this step would not have been added to V in `alterS`.

Hence, there are four possibilities why a step $s' = (z', t_k, z'')$ is not added to S in line 6:

1. there is no path of the form s',
2. z'' does not satisfy the constraints of lines 3 and 4 of `possibilitySplit`,
3. z'' does not satisfy the constraint of line 1 of `costSplit` or

4. $z' = z'' = z^n$.

If Condition 1 holds then Proposition 2 obviously holds for paths containing s' because if there are no paths of the form s' then there are also no dominant paths containing this step. Condition 2 is similar, in lines 3 and 4 of `possibilitySplit` we impose that firing t_j (the transition of the previously considered step s) a number of y times results in a transition from z'' to z^d; if this is not the case, then the path is not of a form that contains s. Hence, there exist no paths of the form (z', t_k, z'', t_j, z^d), and hence no dominant path can have a subpath of this form.

If Condition 3 holds, then there already is a path of another path form whose steps have been considered and which gives cost equal to or lower than the paths that correspond to a form beginning with s. Hence, s' need not be considered.

If Condition 4 holds, then there exists a path with transitions t_k being fired '*within*' z^n. Remember that cost function d_{z^n} has been assigned to z^n in line 8 of `possibiliySplit`, and that this distance corresponds to the path form θ_{z^n}. Hence, in a path corresponding to a path form that meets Condition 4 there exists a subpath that begins in a state $\vec{x} \in \mathcal{X}_{z^n}$ and after y firings of t_k ends in another state $\vec{x}' \in \mathcal{X}_{z^n}$, after which a path of the form θ_{z^n} is taken to the goal set. The path form d_{z^n} is also possible from $\vec{x}$, and this path meets Conditions 4, so the dominant path involving a number of firings of t_k must be better than the path of this form for the proposition not to hold. The cost of y firings of t_k is $y\kappa_k(z^n)$ and the difference between the linear distance function evaluations in the two states is $d_{z^n}(\vec{x}) - d_{z^n}(\vec{x}')$. If this is indeed the case, it holds that it is optimal to fire t_k as often as possible — i.e., until firing t_j would lead to a state outside z^n. Writing $d_{z^n}(\vec{v}) = \vec{\alpha}\vec{v} + \beta$ we must have that

$$\begin{aligned} y\kappa_k(z^n) < d_{z^n}(\vec{x}) - d_{z^n}(\vec{x}') &= d_{z^n}(\vec{x}) - d_{z^n}(\vec{x} + y\vec{u}_k) \\ &= \vec{\alpha}\vec{v} + \beta - \vec{\alpha}(\vec{v} + y\vec{u}_k) + \beta = -y\vec{\alpha}\vec{u}_k \end{aligned}$$

so that in the non-trivial case $y > 0$ it must hold that

$$\kappa_k(z^n) < -\vec{\alpha}\vec{u}_k = d_{z^n}(0) - d_{z^n}(\vec{u}_i)$$

since $d_{z^n}(0) = \beta$. If this is the case, the if-statement in line 9 will evaluate to true, meaning that due to line 10 z^n will be partitioned under a constraint that separates the part where t_k can be fired without leaving z^n and the part where it cannot. Hence, although a path form satisfying 4 will never be added to S in line 6, but a path that fires t_k as much as possible within z^n will be added to S in line 16 and since this is the only way firing t_k within z^n can be dominant, the proposition holds. □

7.6 Conclusions and Discussion

We have presented a novel method to automatically construct a change of measure for speeding up the simulation of rare events in stochastic Petri nets. Our approach

uniquely combines two characteristics: it uses a *high-level description of the model* with much flexibility and expressivity (a stochastic Petri net) and it works *without generating the entire state-space*.

The heart of our method is an algorithm which automatically partitions the state space into a collection of zones. Each zone comprises states in which the shortest paths to the goal set (which can be used to apply ZVA) have a similar form. The zones are demarcated by a set of linear inequalities, thus avoiding enumeration of all states. The number of zones in typical models does not need to increase as the model's size increases.

We have demonstrated that our algorithm works well in two examples. Furthermore, we have proven that when the algorithm terminates, the returned distances of the shortest paths to the goal set are correct. More experimentation will be needed to fully understand its possibilities and limitations and to optimise the implementation, and some extensions of the algorithm are needed to handle certain classes of models. Specifically, three issues remain unresolved. The first is *termination* of the algorithm within finite time. The second is the integrity of the number of steps in a stutter step. The third is the *efficiency* of the resulting importance sampling estimator. For each we give a short discussion.

7.6.1 Termination

If the state space is infinite, it is possible that the (current) algorithm will not terminate. For example, if transition t_1 takes the system closer to the goal states and enables a transition t_2 with a very high firing rate, but firing t_2 disables itself and does not negate the firing of t_1, then a shortest path might alternate between firing t_1 and t_2. This may result in the algorithm constructing an infinite number of zones. A possible solution is to broaden the concept of a stutter step. If a shortest path alternates between a tuple of transitions, the repeated firing of this tuple could be seen as a stutter step in itself, and the sum of the incidence vectors of the individual transitions as the net effect on the marking. Under such a restriction, the space of zones could well be bounded.

7.6.2 Integrality

In line 2 of `possibilitySplit` the function `findNumberOfTransitions` is called, which determines for a step $s = (z, t_i, z')$ the number of firings of t_i needed to go from a state in $\mathcal{X}_z$ to a state in $\mathcal{X}_{z'}$. Depending on (1) the shape of the linear inequality separating z and z', and (2) the incidence vector $\vec{u}_i$, it may happen that y is not necessarily integer-valued for all states in $\mathcal{X}_z$. E.g., in an SPN with two places it may happen for a specific stutter step that `findNumberOfTransitions` returns $2y = x_0 + x_1$, which is fractional if $x_0 + x_1$ is odd. This can be detected when running the algorithm and a naive implementation would simply stop the algorithm without giving an answer. However, a possible solution is to generalise the concept of a zone, allowing constraints involving the modulo-operator. Checking whether

such zones are empty is then no longer an ILP-problem, but can still be solved using branch-and-bound, the most common way of solving ILP-problems.

7.6.3 Importance Sampling Efficiency

The importance sampling measure as defined in (7.2) is inspired by the change of measure proposed in [73], where also the notion of *bounded relative error* comes up. This notion says that as ϵ approaches 0, the ratio of the standard deviation of the estimator to the standard mean remains bounded. This is desirable: since the accuracy of a simulation result is directly linked to this relative error, this means that the time to reach some level of accuracy never crosses a certain threshold value as ϵ becomes smaller.

The authors of [73] show that bounded relative error is guaranteed in their setting under the assumption that the state space is finite and that no *high-probability cycles* exist. Essentially, these assumptions imply that the number of paths ω with $\mathbb{P}(\omega) = \Theta(\epsilon^{d(\vec{x})})$ is finite. If this does not hold, it may be that $\mathbb{P}(\Psi_{\vec{x}}) \neq \Theta(\epsilon^{d(\vec{x})})$.

A possible remedy would then be to perform a loop-detection algorithm on the initial graph returned by Algorithm 7.2 in order to detect the high-probability cycles, and remove them.

CHAPTER 8

Conclusions

In this thesis, we have presented several contributions to the fields of stochastic model checking using simulation. Before we move on to a more detailed discussion, we would like to present a high-level overview of the main contributions.

1. We added to the *understanding* of hypothesis testing methods for statistical model checking. This is particularly relevant because statistical model checking techniques have considerably gained in term of popularity recently, causing the need for clarity to increase.

2. We *automated* the application of rare event simulation techniques for the broad, commonly used modelling class of Markov chains. This is noteworthy because a lack of generality is an issue that troubles many rare event simulation methods.

3. We avoided the *curse of dimensionality* by not needing the full construction of the state space for rare event simulation of stochastic Petri net models. This is interesting because it strikes a good balance between standard Monte Carlo techniques that are unable to exploit information of the model on one hand and numerical techniques that suffer from the state space explosion problem on the other hand.

The outline of the rest of this chapter is as follows. In Section 8.1, we discuss the contributions of this thesis to the field of stochastic model checking using simulation in more detail, while we discuss possibilities for future research in Section 8.2.

8.1 Contributions

We discuss the contributions of each chapter separately.

In **Chapter 2**, we have presented a common framework for comparing testing techniques. Furthermore, we have presented a new test and have discussed how to use this test as a sequential test for evaluating steady-state properties. We have presented an extensive case study involving six tests for statistical model checking, four that are currently implemented in model checking tools, a test from the statistical literature that was not previously used in the field of statistical model checking and our new test.

In **Chapter 3**, we have discussed different asymptotic regimes in the context of birth-death processes, and their influence on the so-called '*dominant paths*' (which are used to determine our change of measure). We have given an overview of commonly found regimes and events, and have discussed their impact on the application of importance sampling. Also, we have proposed a new method for efficiently drawing sojourn times in the regime of large mission times, and demonstrated empirically that our method works well.

In **Chapter 4**, we have studied networks of parallel birth-death processes, where we specifically focused on the Distributed Database System, a well-known case study from the literature that falls into this model category. We have presented a method based on regeneration cycles that performs well even in the regime of fast component repairs. We have generalised this approach to multicomponent systems with the advanced repair strategies of deferred repair and shared repair facilities. We have broadened the concept of a busy cycle, and showed that in the relevant asymptotic our previous analysis still holds.

In **Chapter 5**, we have presented an algorithm for rare event simulation of Markov chains. We focus on the estimation of the probability that some rare state in the Markov chain is visited before some more typical state. The aforementioned algorithm is completely automated and works for a very broad model class. The algorithm produces a change of measure that we proved to have the desirable property of Vanishing Relative Error. The algorithm removes high probability cycles, an obstacle that frequently appears in the literature. We have also generalised this approach to estimate time-bounded versions of this probability in CTMCs. While the method proposed so far makes a vital technical assumption about the system model, we have confidence that this can be overcome in future research.

In **Chapter 6**, we have proposed two new methods that use the information produced by the algorithm and which may lead to variance reduction in addition to the variance reduction achieved by the use of an importance sampling change of measure. These techniques work well if paths in which the rare event occurs through a non-dominant path still appear frequently enough. We have included a detailed case study discussing the performance of these techniques combined with two different common importance sampling measures.

In **Chapter 7**, we have introduced a method to automatically perform rare event simulation in the modelling class of stochastic Petri nets, without generating the entire state space. The main principle that underpins the method is the construction of a so-called zone graph, which is an abstraction of the state space in which each zone corresponds to a set of states of which the paths to the rare set have the same basic form. We have empirically demonstrated the good performance of the method, and proved its correctness.

8.2 Future Work

We have divided the contents of this section into three parts. The first concerns future work in the field of hypothesis testing, found in Section 8.2.1. The second

concerns future work in the field of rare event simulation, found in Section 8.2.2. The third concerns the combination of the two previous subjects, and is found in Section 8.2.3.

8.2.1 Hypothesis Testing

Hypothesis testing is the subject of **Chapter 2**. In general, the class of system models is extremely broad for statistical model checking using hypothesis testing. As mentioned in Section 2.1.1, there are several assumptions, namely: paths being sampled according to a well-defined probability measure, in a finite amount of time (with probability 1) and without encountering non-deterministic choice (as, e.g., in Markov decision processes). Of these three conditions, termination within a finite amount of time is the hardest for the user to check. However, if one restricts the model class to a more restricted setting like stochastic Petri nets, it may be possible to apply a zone-based algorithm similar to the one proposed in Chapter 7. Such an algorithm would work at the level of the model description and would identify regions where the system can get '*stuck*' (i.e., bottom strongly connected components) or infinite-sized regions in which there is a '*drift*' away from the termination states.

Furthermore, each test used to apply statistical model checking depends on user-specified parameters which, depending on the choice of test, can have an impact on any of a test's three main performance criteria: the correctness, the power and the efficiency. These parameters are all related to how far the true probability of interest is away from the boundary value, and perform badly when this distance is different from the one given by the user. With the two tests discussed in Section 2.3, the user can at least be sure that only the efficiency is affected. Still, a fully automated approach would not even require this parameter. Of course, due to its generality, such a fully automated approach would do worse than the most general test so far — namely Darling — when its input parameter γ is chosen just right. This is a trade-off that may be an argument against such a fully generalised approach, since the Darling test is already much less efficient than, e.g., the SPRT, when the parameters are chosen correctly.

To verify steady-state properties using statistical model checking, a large amount of user input is still needed. A major complication occurs when the state space of the model contains a region in which a lot of time is spent in steady-state but which is not visited during 1) a typical renewal cycle or 2) during the '*warm-up*' phase of the simulation. In the first case, a fixed-sample size test using the regenerative approach as in [117] will fail unless the sample size is large enough. In the second case, the sequential test of Section 2.5 that uses batch means will fail unless the batch size is large enough. In both cases, it is left to the user to choose either of these parameters correctly. Again, a possible remedy is to restrict the model class to stochastic Petri nets and to apply an algorithm similar to the one proposed in Chapter 7 to identify these complicated regions (if this is feasbile).

8.2.2 Automated Rare Event Simulation

The classical challenge of rare event simulation (and importance sampling in particular) has long been its generality: approaches that work well in a restricted setting tend to fail when the setting is slightly generalised. Even the identification of rarity regimes, and, by implication, the general behaviour of the set of dominant paths eludes complete generality: the rarity regimes discussed in **Chapter 3** are particularly relevant in a specific model class (the birth-death process) and specific properties (i.e., the '*unbounded until*' property ψ and the '*bounded until*' property $\psi^{\bar{\tau}}$). Typically, it makes sense to divide rarity regimes into '*static*' regimes, in which there is a fixed set of dominant paths that come to dominate the event of interest, and '*dynamic*' regimes, in which this sets changes as the parameters underlying the regime approach their asymptotes. Still, even between the two '*static*' regimes of Section 3.2.1 ('*slow component failures*', $\lambda \downarrow 0$) and of Section 3.2.2 ('*fast component repairs*', $\mu \to \infty$) there is variety in terms of which importance sampling approach works well, although the approach proposed in Chapter 4 works for both of these settings. It would be interesting to see if a general notion of rarity regimes can be given not in terms of parameters in a specific model setting but purely in terms of the behaviour of the set of the dominant paths, which still has the power to describe the structure of a well-performing change of measure.

Despite the fact that the approach of **Chapter 4** works well for several rarity regimes, it is still plagued by importance sampling's classical challenge (i.e., a lack of generality). It works well in its setting (i.e., networks of parallel birth-death processes) but when the setting is slightly generalised it is uncertain whether the approach continues to do so. When we broadened the repair strategy to shared repair facilities in Section 4.3 we did not notice a decrease in performance. However, the effects would be unclear if we added failure propagation (i.e., components of several types failing at the same time) to the model. While the technique may not be general enough to be implemented in common model checking/performance evaluation tools such as PRISM, the insights gained in Chapter 4 can be applied to other settings. Furthermore, the model class, while restricted, is still a relevant model for mass data storage systems and the technique of Chapter 4 can be used to evaluate their performance.

The approach of **Chapter 5** is general enough to be called model checking: it can handle '*unbounded until*' type properties (reaching a rare set before a more typical set) in any Markov chain and '*bounded until*' type properties for CTMCs (reaching a rare set before a more typical set and within a fixed time frame) as long as the time interval is of the form $[0, t]$, $t \in \mathbb{R}^+$. Since evaluation of the next-operator using statistical model checking is largely trivial, only bounded until for DTMCs (i.e., reaching a rare set before a more typical set and before a within a fixed frame of step numbers) and steady-state properties are left as a challenge before the algorithm can handle the full breadth of PCTL- and CSL-model checking (apart from nested operators). For steady-state properties, we can use an approach based on renewal theory similar to the one described in Chapter 4. If the regeneration state is contained in a high-probability cycle, then we may use generalised busy cycles

as discussed in Section 4.2. Also, we can use the results of Chapter 4 to generalise the approach of Chapter 5 to settings with large time horizons (i.e., where the time horizon is inversely proportional to the rarity parameter ϵ): this would yield a time-dependent change of measure for estimating '*bounded until*' type properties.

The technique based on formula (6.6) in **Chapter 6** is useful to supplement the technique of Chapter 5. Although the resulting estimate only works well if non-dominant paths satisfying the event of interest are sampled often enough, an important quality is that if this turns out *not* to be the case after having drawn a sample, we can still use this sample to obtain the standard estimate. To make the technique more broadly applicable (i.e., not just as a supplement to the method of Chapter 5), one could look at ways in which the probability contribution of certain (not necessarily *all*) dominant paths can easily be computed, possibly using high-level analysis as in Chapter 7, and then use this information to achieve variance reduction.

The zone-based approach of **Chapter 7** still has several issues (as discussed in Section 7.6) that either need to be resolved in future research or which require user insight to detect. Apart from these issues, obvious extensions include a generalisation to transient properties (based on the insights of Section 5.5), a dedicated ILP-solver (since the very general one used for our implementation is very slow considering that our problems have a common structure) and an extension to other rarity regimes. An obvious rarity regime is the one of Section 3.2.3 (many spares, $n \to \infty$). This regime is interesting for several reasons: 1) it includes many biochemical models that are increasingly analysed using model checking tools, 2) the state space explosion problem has particularly profound consequences in this regime, and 3) it is not necessary to use an ILP-solver in this setting because one can assume that each zone is non-empty simply because there are so many states. As for the change of measure in this setting, one could think of an approach in which different '*drifts*' are assigned to each of the zones, combined with a '*mollification*' (as in [35]) that is particularly relevant on the boundaries between the zones.

8.2.3 Combining Hypothesis Testing and Rare Event Simulation

Last but not least, there is the unresolved question of how to combine hypothesis testing and importance sampling, as discussed in Section 2.6.1. An interesting question is whether one could use the algorithm of Chapter 5 to construct a change of measure in which the likelihood ratios are bounded from above. When the likelihood ratios are upper bounded, we can use the Chernoff-Hoefding inequality to construct confidence intervals or to apply a fixed sample size test, and we can apply the Azuma test for sequential hypothesis testing. A way to achieve bounded likelihood ratios could be to '*turn off*' the importance sampling when, during the simulation, the likelihood ratio reaches a certain level, the same way we now turn off the importance sampling when we leave Λ, the set of relevant states. It remains to be seen whether we retain important properties such as BRE and VRE in this setting. As mentioned in Section 2.6.1, this idea has not been fully explored, hence, remains a subject for further research.

Bibliography

[1] E. Ábrahám, N. Jansen, R. Wimmer, J. P. Katoen, and B. Becker. DTMC model checking by SCC reduction. In *Proceedings of the Seventh International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 37–46. IEEE, 2010.

[2] M. Ajmone Marsan, G. Balbo, S. Donatelli, G. Franceschinis, and G. Conte. *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, 1994.

[3] S. Amari and R. Misra. Closed-form expressions for distribution of sum of exponential random variables. *IEEE Transactions on Reliability*, 46(4):519–522, 1997.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic (TOCL)*, 1(1):162–170, 2000.

[5] C. Baier, L. Cloth, B. R. Haverkort, H. Hermanns, and J. P. Katoen. Performability assessment by model checking of Markov reward models. *Formal Methods in System Design*, 36(1):1–36, 2010.

[6] C. Baier, P. D'Argenio, and M. Groesser. Partial order reduction for probabilistic branching time. *Electronic Notes in Theoretical Computer Science*, 2006.

[7] C. Baier, B. R. Haverkort, H. Hermanns, and J. P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer Aided Verification*, volume 1855, pages 358–372. LNCS Volume 1855, Springer, 2000.

[8] C. Baier, B. R. Haverkort, H. Hermanns, and J. P. Katoen. On the logical characterisation of performability properties. In *Automata, Languages and Programming*, pages 780–792. LNCS Volume 1853, Springer, 2000.

[9] C. Baier, B. R. Haverkort, H. Hermanns, and J. P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[10] C. Baier and J. P. Katoen. *Principles of model checking*. MIT press, 2008.

[11] B. Barbot, S. Haddad, and C. Picaronny. Coupling and importance sampling for statistical model checking. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 331–346. LNCS Volume 7214, Springer, 2012.

[12] J. Barnat, J. Chaloupka, and J. van de Pol. Distributed algorithms for SCC decomposition. *Journal of Logic and Computation*, 21(1):23–44, 2011.

[13] S. Blom and J. van de Pol. State space reduction by proving confluence. In *Computer Aided Verification*, pages 596–609. LNCS Volume 2404, Springer, 2002.

[14] A. Blum, A. Goyal, P. Heidelberger, S. Lavenberg, M. Nakayama, and P. Shahabuddin. Modeling and analysis of system dependability using the system availability estimator. In *Twenty-Fourth International Symposium on Fault-Tolerant Computing*, pages 137–141. IEEE, 1994.

[15] A. Bobbio and K. S. Trivedi. An aggregation technique for the transient analysis of stiff Markov chains. *IEEE Transactions on Computers*, 100(9):803–814, 1986.

[16] H. Boudali, P. Crouzen, B. R. Haverkort, M. Kuntz, and M. Stoelinga. Arcade - A formal, extensible, model-based dependability evaluation framework. In *13th IEEE International Conference on Engineering of Complex Computer Systems, Belfast*, volume 3, pages 243–248. IEEE Press, 2008.

[17] M. Capiński and P. E. Kopp. *Measure, integral and probability*. Springer, 2004.

[18] J. Carrasco. Failure distance based simulation of repairable fault-tolerant systems. In *Proceedings of the 5th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, pages 351–365, 1992.

[19] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[20] E. Chong and S. Żak. *An Introduction to Optimization*. John Wiley & Sons, 2004.

[21] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. Doyle, W. Sanders, and P. Webster. The Möbius modeling tool. In *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*. IEEE, 2001.

[22] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[23] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Progress on the state explosion problem in model checking. In *Informatics: 10 Years Back, 10 Years Ahead*, pages 176–194. LNCS Volume 2000, Springer, 2001.

[24] E. Clarke, O. Grumberg, M. Minea, and D. Peled. State space reduction using partial order techniques. *International Journal on Software Tools for Technology Transfer*, 2(3):279–287, 1999.

[25] E. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT press, 1999.

[26] L. Cloth and B. R. Haverkort. Five performability algorithms: A comparison. In *MAM 2006: Markov Anniversary Meeting, Charleston, SC, USA*, pages 39–54, Raleigh, NC, USA, June 2006. Boson Books.

[27] D. Darling and H. Robbins. Iterated logarithm inequalities. *Proceedings of the National Academy of Sciences of the United States of America*, 57(5):1188–1192, 1967.

[28] D. Darling and H. Robbins. Some nonparametric sequential tests with power one. *Proceedings of the National Academy of Sciences of the United States of America*, 61(3):804–809, 1968.

[29] A. David, K. Larsen, A. Legay, M. Mikučionis, D. Poulsen, J. Van Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. In *Proceedings of FORMATS 2011*, pages 80–96. LNCS Volume 6919, Springer, 2011.

[30] P. T. de Boer. *Analysis and efficient simulation of queueing models of telecommunication systems*. PhD thesis, University of Twente, 2000.

[31] P. T. de Boer, P. L'Ecuyer, G. Rubino, and B. Tuffin. Estimating the probability of a rare event over a finite time horizon. In *Proceedings of the 2007 Winter Simulation Conference*, pages 403–411. IEEE Press, 2007.

[32] S. Derisavi, H. Hermanns, and W. Sanders. Optimal state-space lumping in Markov chains. *Information Processing Letters*, 87(6):309–315, 2003.

[33] M. Devetsikiotis and J. Townsend. An algorithmic approach to the optimization of importance sampling parameters in digital communication system simulation. *IEEE Transactions on Communications*, 41(10):1464–1473, 1993.

[34] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[35] P. Dupuis, A. D. Sezer, and H. Wang. Dynamic importance sampling for queueing networks. *The Annals of Applied Probability*, pages 1306–1346, 2007.

[36] D. El Rabih, G. Gorgo, N. Pekergin, and J.-M. Vincent. Steady-state property verification: a comparison study. In *Proceedings of the Fourth international conference on Verification and Evaluation of Computer and Communication Systems*, pages 95–106. British Computer Society, 2010.

[37] D. El Rabih and N. Pekergin. Statistical model checking using perfect simulation. In *Automated Technology for Verification and Analysis*, pages 120–134. LNCS Volume 5799, Springer, 2009.

[38] P. H. Feiler, D. P. Gluch, and J. J. Hudak. The architecture analysis & design language (AADL): An introduction. Technical report, Carnegie Mellon University, 2006.

[39] R. M. Feldman and C. Valdez-Flores. *Applied probability and stochastic processes*. Springer, 2010.

[40] G. Fishman. *Discrete-event simulation: modeling, programming, and analysis*. Springer, 2001.

[41] B. Fox and P. Glynn. Discrete-time conversion for simulating finite-horizon Markov processes. *SIAM Journal on Applied Mathematics*, pages 1457–1473, 1990.

[42] P. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, 1989.

[43] A. Goyal, W. Carter, E. de Souza e Silva, S. Lavenberg, and K. Trivedi. The system availability estimator. In *Proceedings of the 16th Int. Symp. on Fault-Tolerant Computing*, pages 84–89, 1986.

[44] A. Goyal, S. Lavenberg, and K. Trivedi. Probabilistic modeling of computer system availability. *Annals of Operations Research*, 8(1):285–306, 1987.

[45] A. Goyal, P. Shahabuddin, P. Heidelberger, V. Nicola, and P. Glynn. A unified framework for simulating Markovian models of highly dependable systems. *IEEE Transactions on Computers*, 41(1):36–51, 1992.

[46] D. Gross, J. Shortle, J. Thompson, and C. Harris. *Fundamentals of queueing theory*. John Wiley & Sons, 1985.

[47] P. Haas and G. Shedler. Stochastic Petri net representation of discrete event simulations. *Software Engineering, IEEE Transactions on*, 15(4):381–393, 1989.

[48] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6(5):512–535, 1994.

[49] B. R. Haverkort. *Performance of computer communication systems: a model-based approach*. John Wiley & Sons, 1998.

[50] P. Heidelberger. Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 5(1):43–85, 1995.

[51] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. *Lecture notes in computer science*, 2937:307–329, 2004.

[52] T. Hill and P. Lewicki. *Statistics: methods and applications: a comprehensive reference for science, industry, and data mining*. StatSoft, 2006.

[53] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, pages 13–30, 1963.

[54] H. Jeffreys. *Theory of Probability*. Oxford University Press, 1961.

[55] C. Jegourel, A. Legay, and S. Sedwards. Cross-entropy optimisation of importance sampling parameters for statistical model checking. In *Computer Aided Verification*, pages 327–342. LNCS Volume 7358, Springer, 2012.

[56] C. Jegourel, A. Legay, and S. Sedwards. A platform for high performance statistical model checking – PLASMA. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 498–503, 2012.

[57] S. Jha, E. Clarke, C. Langmead, A. Legay, A. Platzer, and P. Zuliani. A Bayesian approach to model checking biological systems. In *Computational Methods in Systems Biology*, pages 218–234. Springer, 2009.

[58] J. Júlvez. Basic qualitative properties of Petri nets with multi-guarded transitions. In *American Control Conference*. IEEE Press, 2009.

[59] S. Juneja. Estimating tail probabilities of heavy tailed distributions with asymptotically zero relative error. *Queueing Systems*, 57(2):115–127, 2007.

[60] S. Juneja and P. Shahabuddin. Fast simulation of Markov chains with small transition probabilities. *Management Science*, pages 547–562, 2001.

[61] S. Juneja and P. Shahabuddin. Splitting-based importance-sampling algorithm for fast simulation of Markov reliability models with general repair-policies. *IEEE Transactions on Reliability*, 50(3):235–245, 2001.

[62] S. Juneja and P. Shahabuddin. Rare event simulation techniques: An introduction and recent advances. *Simulation*, pages 291–350, 2006.

[63] J. P. Katoen, M. Khattri, and I. Zapreev. A Markov reward model checker. In *Second International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 243–244. IEEE, 2005.

[64] J. P. Katoen and I. Zapreev. Simulation-based CTMC model checking: An empirical evaluation. In *Sixth International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 31–40. IEEE, 2009.

[65] C. Kelling. A framework for rare event simulation of stochastic Petri nets using "RESTART". In *Proceedings of the 28th Winter Simulation Conference*, pages 317–324. IEEE Computer Society, 1996.

[66] O. Krafft and N. Schmitz. A note on Hoeffding's inequality. *Journal of the American Statistical Association*, pages 907–912, 1969.

[67] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Computer Performance Evaluation: Modelling Techniques and Tools*, pages 113–140. LNCS Volume 2324, Springer, 2002.

[68] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.

[69] T. L. Lai. Sequential analysis: some classical problems and new challenges. *Statistica Sinica*, 11(2):303–350, 2001.

[70] A. Law and W. Kelton. *Simulation modeling and analysis*. McGraw-Hill New York, 1991.

[71] P. L'Ecuyer, J. Blanchet, B. Tuffin, and P. Glynn. Asymptotic robustness of estimators in rare-event simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(1):6, 2010.

[72] P. L'Ecuyer and B. Tuffin. Approximate zero-variance simulation. In *Proceedings of the 2008 Winter Simulation Conference*, pages 170–181. IEEE Press, 2008.

[73] P. L'Ecuyer and B. Tuffin. Approximating zero-variance importance sampling in a reliability setting. *Annals of Operations Research*, 189(1):277–297, 2011.

[74] H. H. Liu. *Software performance and scalability: a quantitative approach*. John Wiley & Sons, 2011.

[75] S. Luke. *Fast Mersenne Twister*. October 2004, http://www.cs.gmu.edu/∼sean/research/mersenne/MersenneTwisterFast.java.

[76] K. Matthes. Zur Theorie der Bedienungsprozesse. In *Proceeding of the Third Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 513–528. Publishing House of the Czechoslovak Academy of Sciences, 1962.

[77] K. L. McMillan. *Symbolic model checking*. Springer, 1993.

[78] D. Miretskiy, W. Scheinhardt, and M. Mandjes. On efficiency of multilevel splitting. *Communications in Statistics – Simulation and Computation*, 41(6):890–904, 2012.

[79] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[80] M. Nakayama and P. Shahabuddin. Quick simulation methods for estimating the unreliability of regenerative models of large, highly reliable systems. *Probability in the Engineering and Informational Sciences*, 18(03):339–368, 2004.

[81] V. Nicola, P. Shahabuddin, and M. Nakayama. Techniques for fast simulation of models of highly dependable systems. *IEEE Transactions on Reliability*, 50(3):246–264, 2001.

[82] A. S. Novozhilov, G. P. Karev, and E. V. Koonin. Biological applications of the theory of birth-and-death processes. *Briefings in bioinformatics*, 7(1):70–85, 2006.

[83] W. Obal and W. Sanders. An environment for importance sampling based on stochastic activity networks. In *Proceedings of the 13th Symposium on Reliable Distributed Systems*, pages 64–73. IEEE, 1994.

[84] S. Parekh and J. Walrand. A quick simulation method for excessive backlogs in networks of queues. *IEEE Transactions on Automatic Control*, 34(1):54–66, 1989.

[85] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57. IEEE, 1977.

[86] J. Propp and D. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, 1996.

[87] D. Reijsbergen, P. T. Boer, and W. Scheinhardt. Transient behaviour in highly dependable Markovian systems: New regimes, multiple paths. In *Proceedings of RESIM 2010*, pages 26–27.

[88] D. Reijsbergen, P. de Boer, W. Scheinhardt, and B. R. Haverkort. Rare event simulation for highly dependable systems with fast repairs. In *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 251–260. IEEE, 2010.

[89] D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and B. R. Haverkort. Fast simulation for slow paths in Markov models. In *Proceedings of RESIM 2012*, pages 36–38.

[90] D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and B. R. Haverkort. On hypothesis testing for statistical model checking. SMC Workshop 2013 (to appear).

[91] D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and B. R. Haverkort. Recent advances in importance sampling for statistical model checking. SMC Workshop 2013 (to appear).

[92] D. Reijsbergen, P.-T. De Boer, W. Scheinhardt, and B. R. Haverkort. Rare event simulation for highly dependable systems with fast repairs. *Performance Evaluation*, 69(7):336–355, 2012.

[93] D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and B. R. Haverkort. Automated rare event simulation for stochastic petri nets. In *Proceedings of the 10th International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 372–388. Springer, 2013.

[94] D. Reijsbergen, P. T. de Boer, W. Scheinhardt, and S. Juneja. Some advances in importance sampling of reliability models based on zero variance approximation. In *Proceedings of RESIM 2012*, pages 30–35.

[95] A. Remke. *Model checking structured infinite Markov chains*. PhD thesis, University of Twente, 2008.

[96] A. Ridder. Importance sampling simulations of Markovian reliability systems using cross-entropy. *Annals of Operations Research*, 134(1):119–136, 2005.

[97] B. Ripley. *Stochastic Simulation*. John Wiley & Sons, 1987.

[98] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis II. System level concurrency control for distributed database systems. *ACM Transactions on Database Systems (TODS)*, 3(2):178–198, 1978.

[99] S. Ross. *Stochastic Processes*. John Wiley & Sons, 1996.

[100] R. Rubinstein and D. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.

[101] A.-E. Rugina, K. Kanoun, and M. Kaâniche. A system dependability modeling framework using AADL and GSPNs. In *Architecting Dependable Systems IV*, pages 14–38. LNCS Volume 4615, Springer, 2007.

[102] W. Sanders and L. Malhis. Dependability evaluation using composed SAN-based reward models. *Journal of parallel and distributed computing*, 15(3):238–254, 1992.

[103] W. Sanders and J. Meyer. Stochastic activity networks: Formal definitions and concepts. *Lectures on Formal Methods and Performance Analysis*, 2001.

[104] I. Sason. Moderate deviations analysis of binary hypothesis testing. *arXiv:1111.1995*, 2011.

[105] I. Sason. On refined versions of the Azuma-Hoeffding inequality with applications in information theory. *arXiv:1111.1977*, 2011.

[106] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *Computer Aided Verification*, pages 202–215. LNCS Volume 3114, Springer, 2004.

[107] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *Computer Aided Verification*, pages 266–280. LNCS Volume 3576, Springer, 2005.

[108] P. Shahabuddin. Importance sampling for the simulation of highly reliable Markovian systems. *Management Science*, pages 333–352, 1994.

[109] L. Shepp. A first passage problem for the Wiener process. *The Annals of Mathematical Statistics*, 38(6):1912–1914, 1967.

[110] B. Tuffin and K. Trivedi. Implementation of importance splitting techniques in stochastic Petri net package. *Computer Performance Evaluation. Modelling Techniques and Tools*, 1786:216–229, 2000.

[111] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[112] H. Younes, E. Clarke, and P. Zuliani. Statistical verification of probabilistic properties with unbounded until. *Formal Methods: Foundations and Applications*, pages 144–160, 2011.

[113] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking: An empirical study. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 46–60, 2004.

[114] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(3):216–228, 2006.

[115] H. Younes and R. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Computer Aided Verification*, pages 223–235. LNCS Volume 2404, Springer, 2002.

[116] H. Younes and R. Simmons. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proceedings of the National Conference on Artificial Intelligence*, pages 742–747. AAAI Press, 2004.

[117] I. Zapreev. *Model checking Markov chains: techniques and tools*. PhD thesis, University of Twente, 2008.

[118] A. Zimmermann, J. Freiheit, R. German, and G. Hommel. Petri net modelling and performability evaluation with TimeNET 3.0. *Computer Performance Evaluation. Modelling Techniques and Tools*, 1786:188–202, 2000.

[119] P. Zuliani, A. Platzer, and E. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 243–252. ACM, 2010.

List of Acronyms

AADL Architecture Analysis & Design Language. 7, 135

BFB Balanced Failure Biasing. 15, 17, 79–84, 91, 93, 94, 97, 98, 121, 124, 126–133, 147–149

BRE Bounded Relative Error. 15–17, 64, 69, 97, 98, 101, 112, 115, 117, 139, 161

CI confidence interval. 21, 28, 29, 47, 50–52, 69–71, 129, 131, 133, 148, 149

CLT Central Limit Theorem. 21, 28, 44, 52

CSL Continuous Stochastic Logic. 3, 7, 8, 10, 12, 23, 49, 57, 58, 160

CTL Computation Tree Logic. 3–8

CTMC Continuous-Time Markov Chain. 6–10, 12, 56, 65, 66, 69, 74, 89, 94, 97, 99, 114–116, 119, 158

DDS Distributed Database System. 73–75

DTMC Discrete-Time Markov Chain. 6, 8–10, 12, 58, 97–99, 114–116, 130, 132

FCFS First Come First Serve. 73, 93, 94

ILP Integer Linear Programming. 143, 155, 161

IS Importance Sampling. 14–16, 53, 69–71, 79–84, 89, 91, 93–96, 121, 126, 130, 134, 147–149

LST Laplace-Stieltjes Transform. 86

LTS Labelled Transition System. 4, 6, 8

MC Monte Carlo. 11, 15, 16, 69–71, 80–83, 91, 93, 94, 129, 130, 147–149

PCTL Probabilistic CTL. 3, 6–8, 10, 12, 23, 49, 160

SPN Stochastic Petri Net. 7–9, 14, 17, 19, 57, 73–75, 99, 135–137, 154

SPRT Sequential Probability Ratio Test. 27, 29–32, 35, 36, 44–53, 159

VRE Vanishing Relative Error. 15, 17, 98, 99, 112–115, 161

ZVA Zero Variance Approximation. 15, 18, 98, 100, 121, 124, 126, 128–131, 133, 134, 138, 154

About the Author

Daniël Reijsbergen was born in The Hague, the Netherlands, on 10 September 1985 and grew up in the charming Laakkwartier district of the city. After obtaining his *gymnasiumdiploma* in 2003 he studied Econometrics and Operations Research at the Vrije Universiteit in Amsterdam, obtaining his B.Sc. degree in 2006 and his M.Sc. degree in 2009. Concurrently, he completed the first year of the Arabic Language and Culture Bachelor's programme at the University of Amsterdam and spent four months at the Netherlands institute in Cairo. He began his work as a Ph.D. candidate at the DACS and SOR groups of the University of Twente shortly after returning to the Netherlands, resulting in this thesis. He is currently employed as a postdoctoral researcher at the University of Edinburgh. His research in Edinburgh is part of the QUANTICOL project, and presently focuses on the modelling of bus movements.

The following is a list of his publications in reverse chronological order:

- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "On Hypothesis Testing for Statistical Model Checking," SMC Workshop 2013.
- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Recent Advances in Importance Sampling for Statistical Model Checking ," SMC Workshop 2013.
- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Automated rare event simulation for stochastic petri nets," in *Proceedings of the 10th International Conference on the Quantitative Evaluation of Systems (QEST)*, 2013.
- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and S. Juneja, "Some advances in importance sampling of reliability models based on zero variance approximation," RESIM 2012.
- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Fast simulation for slow paths in Markov models," RESIM 2012.
- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," *Performance Evaluation*, vol. 69, no. 7, pp. 336–355, 2012.
- D. Reijsbergen, P.T. de Boer, W. Scheinhardt, and B.R. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," in *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, 2010, pp. 251–260.
- D. Reijsbergen, P.T. de Boer, and W. Scheinhardt, "Transient behaviour in highly dependable Markovian systems: New regimes, multiple paths," RESIM 2010.

Acknowledgements

Although the title page of this thesis mentions only a single name, the four-year process of creating it is by no means individual. In particular, the ideas contained within its pages have been refined and in some cases conceived during many discussions with others. Without my supervisors, Pieter-Tjerk de Boer and Werner Scheinhardt, and the many meetings and conversations that we have had regarding my research over the years, this thesis would be a mere shell of its present form. My other co-authors, Boudewijn Haverkort and Sandeep Juneja, also deserve specific mention in this context. The initial motivation for the research underlying Chapter 2 was a discussion with Boudewijn, while discussions as part of a research visit by Sandeep form the basis of Chapter 5 and particularly Chapter 6. I would also like to specifically thank Boudewijn for making me aware of the job vacancy that led to my current employment. I would like to thank Richard Boucherie for his input regarding this thesis, especially regarding the structure. Furthermore, I would like to thank all the other committee members for reading this thesis and for their constructive feedback. I would in particular like to thank Ad Ridder for introducing me to rare event simulation and putting me in contact with Pieter-Tjerk and Werner about the position in Twente that I would come to fill. Credit is also due to my (co-)promotors for writing the research proposal that led to this thesis, and to NWO (and, by extension, the Dutch taxpayer) for funding it.

I would also like to thank all the staff at the DACS and SOR groups for making work enjoyable. A few of them deserve particular mention. Stephan, Hamed and Jasper for being fun officemates; Martijn for the many lunch walks and for heroically saving my bicycle at Hengelo station; Anja for being a fun travelling companion for the ROCKS meetings; Yanting for being both a fun and reliable assignment partner for the LNMB courses and Mihaela for the many lunch sessions. I would also like to thank Anne Remke and Dick Meijer for the good cooperation on the coursework in which I was involved.

Finally, I would of course like to thank my family and friends (which again includes Martijn and Stephan) for all the time spent outside working hours. Your impact on the contents of this thesis may be less visible, but it is certainly present.