Tools at the Frontiers of Quantitative Verification[⋆]

QComp 2023 Competition Report

Roman Andriushchenko[1], Alexander Bork[2], Carlos E. Budde[3], Milan Češka[1], Kush Grover[4], Ernst Moritz Hahn[5], Arnd Hartmanns[5]✉, Bryant Israelsen[6], Nils Jansen[7], Joshua Jeppson[6], Sebastian Junges[7], Maximilian A. Köhl[8], Bettina Könighofer[9], Jan Křetínský[4,10], Tobias Meggendorfer[4,11,12], David Parker[13], Stefan Pranger[9], Tim Quatmann[2], Enno Ruijters, Landon Taylor[6], Matthias Volk[14], Maximilian Weininger[4,11], and Zhen Zhang[6]

[1] Brno University of Technology, Brno, Czech Republic
[2] RWTH Aachen University, Aachen, Germany
[3] University of Trento, Trento, Italy
[4] Technical University of Munich, Munich, Germany
[5] University of Twente, Enschede, The Netherlands
[6] Utah State University, Logan, Utah, USA
[7] Radboud University, Nijmegen, The Netherlands
[8] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[9] Graz University of Technology, Graz, Austria
[10] Masaryk University, Brno, Czech Republic
[11] Institute of Science and Technology Austria, Klosterneuburg, Austria
[12] Lancaster University Leipzig, Leipzig, Germany
[13] University of Oxford, Oxford, UK
[14] Eindhoven University of Technology, Einhoven, The Netherlands
✉ a.hartmanns@utwente.nl

Abstract. The analysis of formal models that include quantitative aspects such as timing or probabilistic choices is performed by quantitative verification tools. Broad and mature tool support is available for computing basic properties such as expected rewards on basic models such as Markov chains. Previous editions of QComp, the comparison of tools for the analysis of quantitative formal models, focused on this setting. Many application scenarios, however, require more advanced property types such as LTL and parameter synthesis queries as well as advanced models like stochastic games and partially observable MDPs. For these, tool support is in its infancy today. This paper presents the outcomes of QComp 2023: a survey of the state of the art in quantitative verification

[⋆] The authors are ordered alphabetically. This work was supported by DFG RTG 2236/2 (UnRAVeL) and DFG project TRR 248 (CPEC, ID 389792660), by the EU under MSCA grant agreements 101008233 (MISSION), 101034413 (IST-BRIDGE), and 101067199 (ProSVED), by ERC Starting Grant 101077178 (DEUCE), ERC Consolidator Grant 864075 (CAESAR), and ERC Advanced Grant 834115 (FUN2MODEL), by GAČR grant GA23-06963S (VESCAA), by National Science Foundation grant 1856733, by NextGenerationEU project D53D23008400006 (SMARTITUDE), and by NWO VENI grant 639.021.754.

tool support for advanced property types and models. With tools ranging from first research prototypes to well-supported integrations into established toolsets, this report highlights today's active areas and tomorrow's challenges in tool-focused research for quantitative verification.

1 Introduction

The inclusion of quantitative aspects such as probabilistic choices, timing, and random delays in system modelling is crucial to ensure the correctness, performance, and dependability of the ever-increasing amount of complex safety- and economically-critical systems that support our societies. Well-known examples include the use of randomised algorithms in Internet protocols to achieve both simplicity and scalability [156] or the fault tree modelling approach for safety assessment in the nuclear industry [94].

Formally, these aspects can be captured in established mathematical **formalisms** like discrete- and continuous-time Markov chains (DTMCs and CTMCs) for probabilistic choices and stochastic timing, or more recent notions such as timed automata (TA) [4] for real-time behaviour. Combining DTMCs with nondeterministic (i.e. unquantified and controllable or adversarial) choices results in the nowadays-popular formalism of Markov decision processes (MDPs) [29,194]. These form the mathematical foundation of *quantitative modelling*; for practical purposes, models are specified in a higher-level **modelling language**—such as Modest [30,105] or the Prism language [163]—that is equipped with a semantics in terms of one of the formalisms. When combined with a query for a numerical *property* of a model, e.g. for the probability of reaching a set of undesirable states or for the expected reward until a terminal state is reached, we have a basic *quantitative verification* problem.

Basic Quantitative Verification Comparisons

The *basic problems*—i.e. computing a (i) reachability probability, (ii) expected accumulated reward, or (iii) steady-state probability on a DTMC, CTMC, or MDP model[15]—can be solved by various software tools developed over the past two decades. Most tools use one of two approaches: either probabilistic model checking (PMC) [20,116], which applies a numeric algorithm onto a complete in-memory representation of a model's state space, or statistical model checking (SMC) [3,169,221], which randomly samples (or: *simulates*) and statistically analyses a set of model behaviours; or a hybrid approach combining aspects of PMC and SMC such as partial exploration [148], probabilistic planning [142,176], (deep) reinforcement learning [36,98], or Monte Carlo tree search [11].

[15] Probabilistic timed automata (PTA) [166] can be turned into equivalent MDP [165, 167] (or be solved as stochastic games [162]), so treat them like MDP here.

The QComp competition. The 2019 Comparison of Tools for the Analysis of Quantitative Formal Models (QComp 2019) [103] compared the performance, versatility, and usability of nine such tools on a benchmark set of 100 basic quantitative verification problems[16]. It was the first tool competition in quantitative verification, part of the TOOLympics at TACAS 2019 [25]. The next edition of QComp in 2020 [46] used the same benchmark set, but focused more specifically on the different types of correctness guarantees provided by the different tools, highlighting the interplay between performance and precision in quantitative verification. The results of QComp 2020 were presented at the ISoLA 2020/2021 conference. Although the main outcomes of these two editions of QComp were performance results, they were meant as *friendly competitions*: We did not establish a ranking of tools or point out a "winner"; rather, we highlighted the capabilities, strengths, and specific niches of all participating tools. In particular, the results clearly showed that some tools were generalists solving many types of problems, while others were specialised to specific tasks where they performed much better than any other participant. The entire performance evaluation and report-writing process was performed in close collaboration with the participants, most of which were the main developers of the respective tools.

Benchmark sets and formats. Aside from providing information about the capabilities and performance of the participating tools, these two editions of QComp also benefited the collaboration and alignment inside the quantitative verification research community: In a parallel effort to QComp 2019, we established the Quantitative Verification Benchmark Set (QVBS) [119], from which the competition selected its 100 benchmark instances. Although the QVBS' models were collected from various sources and came in various modelling languages, the QVBS as a matter of principle includes a translation of each model and its properties into the Jani interchange format [44]; as a result, any tool that supported Jani could participate in QComp 2019 and 2020. Jani thus benefited QComp and the participating tool authors by simplifying frontend development, while QComp furthered the establishment of Jani as a community standard.

QComp 2023: Looking into the Future

While improving solution methods for basic problems remains an active research topic (cf. e.g. [28, 102, 111, 116, 117, 128]), most of today's work in quantitative verification focuses on what we refer to as *advanced problems*: Computing more complex properties on the basic models, computing basic properties on more complex models, or combinations thereof. Most papers include an experimental evaluation, which, however, often uses an ad-hoc research prototype implementation, most of which are *not* further developed into a stable and maintained

[16] In the tool competition context, our verification problems are called *benchmark instances*. Since most benchmark models are parametrised but basic problems ask for a single result value, a benchmark instance is a triple of a model, a concrete parameter valuation, and a property to evaluate. We cover parametric analysis in Sect. 7.

tool. Nevertheless, as QComp 2020 was presented at ISoLA 2020/2021, it became clear that more and more solution methods for advanced problems were being turned into tools of their own or integrated into existing stable tools such as Prism [163] or Storm [125]. Therefore, the next edition of QComp that we present in this report, QComp 2023, shifts its focus towards these *frontiers in quantitative verification.*

Aims. The aims of QComp 2023 are (i) to describe advanced problems in quantitative verification for which analysis algorithms have more recently been developed and first tool support is appearing, (ii) to document the state of the art of this tool support, in terms of what is available today and what pieces are still missing, and (iii) to perform the first comparative tests of these tools where appropriate. The outcome of QComp 2023 is this competition report, which can serve as a guide to state-of-the-art tools for the domain expert faced with an advanced quantitative verification problem, as a historical reference for tool developers, and as a call to action pointing researchers to where better algorithms are still needed and tool developers to where "market opportunities" exist that can be filled with new tools.

Setup and process. QComp 2023 is a more friendly competition than ever: It started with an open call for participation to the quantitative verification community in summer 2022. The interested participants then followed an iterative process of determining *categories* (i.e. advanced problem scenarios) of interest, which included identifying and contacting additional participants. Out of the group of all participants, we then established category coordinators who would lead the process needed to achieve the aims of the competition in their category. As the QComp categories covered all kinds of research and tooling maturity levels, part of the task of the category coordinators was to establish the scope and refine the concrete aims of their category. In categories where several sufficiently stable tools already exist, coordinators could choose to include a performance evaluation, while more cutting-edge categories would focus on a description of the category, available approaches, and prior experimental results if available. The category coordinators delivered the outcomes of their category to the overall QComp 2023 coordinator before summer 2023; over that summer, we integrated all contributions into this report.

In this distributed and flexible approach where the competition is divided into sub-groups that establish the actual aims of their own, QComp 2023 was modelled after the ARCH-COMP friendly competition on verifying continuous and hybrid systems (see cps-vo.org/group/ARCH/FriendlyCompetition), which has been running on this model successfully for seven editions as of today since 2017 [90], with its latest edition concluded just before QComp 2023 this summer.

2 Categories and Participants

As a friendly competition, QComp 2023 was open to all interested parties for suggesting, coordinating, and participating in categories related to quantitative verification. All participants of QComp 2023 are co-authors of this competition report. The competition as a whole was coordinated by A. Hartmanns. Before presenting the results of the individual categories in the remainder of this report, we give an overview of QComp 2023's ten categories with credits to the respective organisers and participants, and present the participating tools.

2.1 Categories

Infinite-state and population models (∞-*state*, Sect. 3): coordinated by Z. Zhang; participants: M. Češka, E. M. Hahn, J. Jeppson.

Long-run average rewards (*LRA*, Sect. 4): coordinated by K. Grover, J. Křetínský, and M. Weininger; participants: A. Hartmanns, T. Meggendorfer, and T. Quatmann.

Linear temporal logic (*LTL*, Sect. 5): coordinated by J. Křetínský and M. Weininger.

Multi-objective analysis (*multi-obj.*, Sect. 6): coordinated by T. Quatmann; participants: K. Grover, D. Parker, and M. Weininger.

Parametric Markov models (*parametric*, Sect. 7): coordinated by S. Junges.

Partially-observable MDPs (*POMDPs*, Sect. 8): coordinated by A. Bork; participants: R. Andriushchenko and D. Parker.

Rare events (*rare events*, Sect. 9): coordinated by C. E. Budde; participants: B. Israelsen, E. Ruijters, L. Taylor, M. Volk, and Z. Zhang.

Robust decision-making under uncertainty (*uncertainty*, Sect. 10): coordinated by N. Jansen; participants: D. Parker.

State space exploration (*exploration*, Sect. 11): coordinated by M. A. Köhl; participants: A. Hartmanns and T. Quatmann.

Stochastic games (*st. games*, Sect. 12): coordinated by D. Parker; participants: B. Könighofer, T. Meggendorfer, S. Pranger, and M. Weininger.

2.2 Participating Tools

Various tools ranging from research prototypes to mature toolsets are available today to tackle the problems covered by the different categories. In Table 1, we list which tools participated in which of the categories of QComp 2023. The meaning of "participate", however, can have a very different meaning in different categories; for example, the *parametric* category only names the four tools that support the analysis of parametric Markov models, while the *multi-obj.* category benchmarks its five participating tools and reports on their relative performance. Categories that include an experimental evaluation such as runtime benchmarking are indicated by a "Y" in the row labelled "*experiments*"; then row "*benchmarks*" states the number of benchmark instances considered in the experimental evaluation[17]. Participation of a tool in any category was volun-

[17] More benchmarks may be *available* for the problems covered by a category, and category *parametric* has no performance evaluation but introduces a benchmark set.

Table 1. Tools participating in QComp 2023's different categories

	∞-state	LRA	LTL	multi-obj.	parametric	POMDPs	rare events	uncertainty	exploration	st. games
experiments	N	Y	N	Y	N	Y	Y	N	Y	Y
benchmarks	–	20	–	66	–	3	10	–	229	16
Dftres							✓			
Epmc			✓	✓	✓					✓
Fig							✓			
Infamy	✓									
mcsta		✓							✓	
modes							✓			
Momba									✓	
MultiGain		✓	✓	✓						
Param					✓					
Paynt						✓				
Pet		✓								✓
Prism			✓	✓	✓	✓		✓		
Prism-games		✓		✓						✓
Ragtimer							✓			
SeQuaiA	✓									
Stamina	✓									
Storm		✓	✓	✓	✓	✓			✓	
StormDftRes							✓			
Tempest		✓								✓

tary and not automatic; in particular, if a tool does not participate in a certain category, this does *not* imply absence of support for the advanced properties or model types that the category focusses on in the tool. To allow the individual category sections to focus on the specifics of their topic, we briefly introduce all 19 tools:

Dftres [48], available at github.com/utwente-fmt/DFTRES, is a statistical model checker designed for repairable dynamic fault trees (DFTs [200]) specified in Galileo and more general CTMCs specified in Jani. It is written in Java and is portable to, at least, Linux, Windows, and macOS.

Epmc [91], available at github.com/iscas-tis/ePMC, is an extensible probabilistic model checking framework mostly written in Java. It is a successor of IscasMC [109].

Fig [43], available at git.cs.famaf.unc.edu.ar/dsg/fig, is a statistical model checker for transient and steady state reachability properties in CTMCs and input/output stochastic automata (IOSA) [71]. Fig is written in C++ and runs on Linux.

Infamy [106], available at depend.cs.uni-saarland.de/tools/infamy, is a tool with the purpose of model checking formulae in continuous stochastic logic (CSL) [14, 21] on infinite-state CTMC specified in a variant of the Prism language by exploring the model up to a certain depth repeatedly. Infamy can also handle certain reward properties.

mcsta, available at modestchecker.net, is the explicit state model checker of the Modest Toolset [113], a collection of tools for the modelling and analysis of stochastic timed and hybrid systems. Its core functionality is the disk-based explicit-state model checking of MDPs [114], MAs [51], PTAs [112], and stochastic timed automata [104]. The Modest Toolset is mainly written in C# and runs on 64-bit Linux, macOS, and Windows systems. It supports the Modest [30, 105] and Jani [44] input languages.

modes [41], available at modestchecker.net, is the Modest Toolset's statistical model checker. It supports the same input languages and platforms as mcsta. It contains simulation engines specialised to different formalisms from DTMCs to stochastic hybrid automata with general probability distributions (SHA) [89], including support for non-linear continuous dynamics [186].

Momba [145], available at momba.dev, is a Python library centred around Jani with the goal of providing easy access to quantitative modelling capabilities.

MultiGain [37] is an extension of Prism for multiple long-run average rewards. MultiGain 2.0 [23], available at zenodo.org/records/10610642, builds on MultiGain, adding support for verification and strategy synthesis for LTL.

Param [107], available at depend.cs.uni-saarland.de/tools/param, was the first tool implementing verification algorithms for parametric Markov models.

Paynt [9], available at github.com/randriu/synthesis, is a tool originally developed for the inductive synthesis of probabilistic programs. It aims at directly synthesising finite-state controllers for partially-observable MDPs.

Pet [179] available at gitlab.lrz.de/i7/partial-exploration, is a model checker focusing on value iteration approaches augmented by partial exploration, based on [36] for reachability with subsequent extensions to mean payoff [12] and cores [148]. It is backed by tailored data structures and algorithms for this purpose, and implemented in Java.

Prism [163], available at prismmodelchecker.org, is a widely-used probabilistic model checker supporting a large range of models and temporal logics. It is a user-friendly tool that comes with a cross-platform graphical user interface. Prism is mostly written in Java, with some algorithms implemented in C.

Prism-games [158], available at prismmodelchecker.org/games, is an extension of Prism focused on the verification of stochastic games.

Ragtimer [129, 211], available at github.com/fluentverification/ragtimer, is designed for chemical reaction networks (CRNs) modeled as CTMCs, combining guided stochastic simulation and commutability properties to compute lower-bound rare event probabilities from a partial state space.

SeQuaiA [54], available at sequaia.model.in.tum.de, offers two powerful engines for the quantitative analysis of population models given as chemical reaction networks via abstraction and simulation. Both build on an interval population abstraction of the underlying CTMC. SeQuaiA comes with a GUI, allowing for convenient modelling and tweaking the models as well as displaying the abstractions and analyses results for better explainability.

Stamina [134,183,184,199], available at staminachecker.org, is an infinite-state PMC tool that iteratively explores a partial state space for a bounded or unbounded CTMC model. The CTMC transient analysis on the partial state space is delegated to Prism's and Storm's PMC engines. Stamina/Prism implements the Stamina 2.0 algorithm and interfaces with Prism's Java API and uses Prism for model parsing and checking. Stamina/Storm is a reimplementation and extension of the Stamina 2.0 algorithm using Storm.

Storm [125], available at stormchecker.org, is a general purpose, high-performance feature-rich probabilistic model checker built around a modular core with an emphasis on time and memory efficiency. Written in C++, Storm's modular design enables the utilization of different model checking engines catering to the characteristics of different models. Notably, Storm excels in efficient symbolic model checking through its `dd` engine leveraging binary decision diagrams (BDDs).

StormDftRes, available at gitlab.utwente.nl/fmt/fault-trees/storm-dft-res, implements multi-threaded Monte Carlo simulation for (non-repairable) DFTs given in either the Galileo or a custom format. StormDftRes builds on the Storm-dft library [217] of Storm, which implements efficient state space generation for DFTs by exploiting e.g. irrelevant failures and symmetries.

Tempest [192] available at tempest-synthesis.org, is based on the Storm model checker, extending its feature set to turn-based stochastic games with a focus on synthesizing most-permissive strategies.

3 Infinite-State and Population Models

In many biochemical reaction and synthetic biology applications, very complex systems are studied and thus software tools become very advantageous and even indispensable for their understanding. For instance, the signalling pathways, chemical reaction networks, and genetic regulatory networks under study consist of many concurrent reactions running at very different speeds and probabilities, with species of both low and high copy numbers. This results in stiff systems suffering from stochasticity/multi-modality and state-space explosion, respectively [95, 212], calling for dedicated analysis tools.

In order to analyse such systems, so-called *population models* are considered. A state of a population model is a tuple of integers, with the i-th component representing the copy number of the i-th species. Hence the state space is typically (countably) infinite. Transitions between states represent executing one reaction of the system. Given that the timing aspect is crucial and that the probability for a reaction to occur is (approximately) exponentially distributed as a function of real time, the model can be defined as a CTMC. This explicit model can be derived directly from a symbolic representations of the system as, say, a *chemical reaction network* (CRN): The rates of the CTMC can be computed from the rates of the CRN reactions and the copy numbers in each state using the mass action kinetics.[18] This transformation immediately enables the applicability of probabilistic model checkers for CTMC to biological systems. However, in order to make the analysis practical, the population structure has to be exploited in dedicated ways. In particular, one has to deal with the huge and in general infinite state spaces.

To handle such state spaces, various **reduction techniques** have been proposed that either truncate states of the underlying CTMC with insignificant probability [182] or leverage structural properties of the CTMC to aggregate/lump selected sets of states [1, 16]. The *interval abstraction* of the species population is a widely used approach to mitigate the state-space explosion problem [223]. Alternatively, several hybrid models have been considered, such as treating only small-population species stochastically while using a deterministic semantics for large-population species [126], applying a moment-based description for medium/high-population species [121], or using the LNA approximation with an adaptive partitioning of the species according to leap conditions [53].

The investigated **properties** range from transient ("What is the (distribution over) states at time t?") to steady-state analysis (concerning the limiting distribution or LRA reward). The typical output of a tool for such a query is either a a certain probability bound or an exact probability (or probability bound) of the predicate being true. Given the numeric character of the results and methods, approximate solutions are considered. Further, in contrast to verification, given that the systems are mostly neither safety-critical, nor completely modelled, it is typically acceptable to produce results without precise error bounds: often by simulation-based techniques [93] or aggressively practical, e.g. semi-quantitative [55], model-based approaches.

3.1 Tool Support and Benchmarks

The main technical characteristics of the available tools participating in this category of QComp 2023 are listed in Table 2.

Infamy model-checks infinite-state CTMC specified in a variant of the Prism language. It is capable of handling the time-bounded subclass of the logic CSL and certain reward properties. It explores the model up to a certain depth repeatedly while descending into the nested CSL formula. Infamy provides different means for finding a stopping criterion for the state-space exploration. This is because there is a trade-off between when to stop and the memory needed to store the finite truncation of the state space.

[18] Consequently, more symbolic models such as stochastic Petri nets are hard to use since the rate of a transition for a particular reaction differs from state to state.

Table 2. Feature comparison of tools for population models

Tool	Platforms	Approach	Models	Syntax	Semantics
Infamy	Linux	model checking + state truncation	CTMC	Prism	CTMC
SeQuaiA	multi-platform	population abstraction + numerical, simulation	population models	GUI, dedicated	CTMC
Stamina	Linux, macOS	model checking + state truncation	CTMC	Prism	CTMC

SeQuaiA offers two engines, both building on a "population" abstraction of the underlying CTMC, abstracting concrete copy numbers to given intervals. The first one [54] computes an abstraction of the CTMC using *acceleration*, abstracting not only states and single transitions, but taking into account sequences of transitions. The resulting model is (i) small enough to *explain* the overall dynamics, and (ii) despite the induced imprecision, allows for a *semi-quantitative analysis*, computing not the exact probabilities of different behaviours, but their orders of magnitude, which is often sufficient in the biological applications. The engine thus features unprecedented scalability, analysing standard complex benchmarks within a fraction of a second, while it is precise enough to conclude on the qualitative behaviour of the system including rare behaviours and on rough estimates of the quantities (population sizes, times). The second engine provides a more precise quantitative analysis by uniquely *combining* the population abstraction with advanced simulation techniques [124]. It is based on a memoization technique that combines previously generated *segments* of runs defined over abstract states to generate new simulations more efficiently while preserving the original system dynamics and its diversity. It adapts online to identify the most important abstract states and thus utilizes the available memory efficiently. In combination with a novel fully automatic and adaptive hybrid simulation scheme, this speeds up the generation of trajectories and correctly predicts the transient behaviour of complex stochastic systems.

Stamina iteratively explores a partial state space where a majority of the probability mass resides. It expands the state space *on the fly* based on the estimated state reachability probability, and truncates a search path when the estimate drops below a user-specified threshold. Stamina then performs time-bounded transient PMC analysis by interfacing with Prism or Storm. In this way, it computes a lower and upper bound, P_{min} and P_{max}, respectively, such that the actual probability of the CSL property under verification lies within $[P_{min}, P_{max}]$. The tightness of the probability window, $w = P_{max} - P_{min}$, is specified by the user, albeit with higher run-time for a smaller w. Stamina can efficiently produce an accurate probability bound for CTMCs with an extremely large or infinite state space. It is not restricted to specific types of input models as long as they can be modelled as CTMCs using the Prism modelling language. Examples include genetic regulatory networks [86, 173], biochemical reaction systems [76, 157], dynamic fault trees (DFTs) [217], and queuing network models [127, 131]. Stamina has been designed to support multiple exploration methods, and can be tailored to the model or property under verification. It has also been designed to be user-friendly and modular. Additionally, a graphical user-interface (written in Qt5) is under active development and will enhance user-experience and ease of use of Stamina.

Benchmarks include the CTMC models in the QVBS, the PRISM Benchmark Suite [164], and the Infamy case studies[19]. In addition, the Stochastic Model Case Studies repository [49][20] hosts a large collection of case studies focusing on biochemical systems with infinite state spaces. Stamina has been evaluated on selected CTMC benchmarks from these benchmark suites, e.g. [134, 199]. SeQuaiA has been evaluated on models describing challenging CRNs from the literature [54].

3.2 Outlook

Aside from the scalability limitation the tools are trying to mitigate, there are specific challenges in the analysis of biochemical and synthetic biological systems.

First, it is a very strong assumption that the correct model is available. Consequently, the analysis methods should be able to effectively handle various forms of **model uncertainty**, including unknown reaction rate parameters, unknown reactants or products, as well as unknown species bounds. The uncertainty can be modelled by various formalisms such as parametric or interval CTMC (see Sect. 10), for which the existing tools offer only very limited support.

Second, **concurrency** is fundamental to these systems, as their constituent chemical reactions are often simultaneously enabled. All enabled concurrent reactions may occur in a state but with very different probabilities, and their noisy operating environment can introduce extremely infrequent but potentially detrimental faults (see Sect. 9). Additionally, their regulatory nature and constituent reversible reactions can cause **cyclic behaviours** and often require long reaction execution sequences to reach a desirable state.

Finally, the verification tools should offer to the users (i.e. biologists) not only the verification result, but also an artifact in the form of a critical sub-system or a critical set of paths allowing the users to **interpret and explain** the results. While some rudimentary effort has been made, e.g. [55], this field is wide open.

4 Long-Run Average Rewards

Many frequently-studied classes of properties of probabilistic systems are based on *rewards*. A reward function assigns to every state (or action or state-action pair) a number modelling a cost (or a payoff) related to the single move. These

[19] https://depend.cs.uni-saarland.de/tools/infamy/casestudies/

[20] https://github.com/fluentverification/CaseStudies_StochasticModelChecking

Table 3. Feature comparison of tools for average-reward properties

Tool	Objective	Model	Guarantees
mcsta	ELRA	CTMC, MA	ε
MultiGain	ELRA, SS	MDP	E-FP
Pet	ELRA	DTMC, CTMC, MDP	ε
Prism-games	ELRA	TSG	none
Storm	ELRA, SS	DTMC, CTMC, MDP, MA	ε, E-RA
Tempest	ELRA	TSG	none

rewards are accumulated over infinite paths in various manners. Popular ways are discounted, total, and average rewards [194]. While the *discounted reward* is heavily used in diverse applications ranging from economics to robotics, and is very easy to optimize, it essentially reflects a limited time horizon only. The *total reward* can reflect longer horizons better (e.g. unbounded reachability), yet not really the infinite-run behaviour. The *average reward* (also known as long-run average reward, limit-average reward, steady-state reward, or mean payoff) captures much more adequately the performance over an unknown or variable horizon (see e.g. [203]). Consequently, it is used to model e.g. performance properties, such as the average delay between requests and responses, the average rate of a particular event, etc. Considering the infinite horizon makes both classic and learning algorithms less efficient. The whole problem is thus more difficult, and also less studied in the context of AI or robotics. In contrast, it is significantly studied in formal verification where performance and dependability are critical and hard guarantees are desirable. Related to the average reward and reducible to it are constraints on the steady state of a system, which become more studied also in the context of AI; see e.g. [146,214]. The algorithms for long-run average (LRA) reward properties again span the whole spectrum of linear and dynamic programming, including value and strategy iteration, with the usual advantages and disadvantages. A specific case is the traditional steady-state analysis on (fully stochastic) Markov chains. There, solving a system of linear equations is sufficient, but for efficiency reasons often replaced by value iteration, too.

4.1 Algorithms and Tool Support

Table 3 gives an overview of tools supporting average-reward properties, differentiating the exact kind of supported objective (SS: steady-state or ELRA: expected long-run average reward), the supported models (where MA are Markov automata [79] and TSG are turn-based stochastic games, see Sect. 12), and the guarantees provided on the precision of the result (either none, ε-precise, E-FP: exact up to floating-point precision, or ERA: exact using rational arithmetic). We complement this high-level overview with short tool descriptions:

mcsta supports [51] model-checking LRA reward properties in MA (and thus also in CTMC as a special case) using either a reduction to linear programming [101] or an ε-precise method based on value iteration [52].

MultiGain implements a linear programming-based approach [35] for multi-objective steady-state and LRA reward objectives in MDP (see also Sect. 6).

Pet focuses on partial-exploration techniques, for which it includes an extension to mean-payoff objectives [12].

Prism-games supports a wide range of zero-sum properties; for LRA rewards (and its multi-objective variant of *ratio objectives*), the stochastic games are required to be controllable multichains, i.e. the sets of states that can occur in any maximal end component must be almost surely reachable.

Storm can answer a wide range of queries for many different models, offering both value iteration-based approaches providing ε-precise results as well as linear programming-based algorithms using exact rational arithmetic. We highlight that Storm is the only tool able to handle negative rewards directly, while others require the rewards to be rescaled first (see Appendix A in the full version of [152] for the standard transformation).

Tempest implements mean-payoff analysis for TSGs on top of Storm, using value iteration with explicit state space representations.

4.2 Performance Comparison

Our performance comparison is only on MDPs, as this is the model that most tools support. Thus, we ran Storm (using the default value iteration which provides ε-guarantees), MultiGain (using linear programming), and Pet (using value iteration and partial exploration). We collected benchmarks from several sources [2, 23, 119]; Appendix G.1 of the full version of [2] contains descriptions of many of them. The experiments were conducted on a freshly installed Ubuntu virtual machine on top of an Intel i7-1165G7 CPU and 8 GB RAM. Each run had access to all eight cores available in the virtual machine and the tools were executed sequentially using a bash script, starting with Storm (on all benchmarks) and ending with Pet. Table 4 shows for every benchmark some characteristics of the model and then the time in seconds taken by each tool (where MG 2.0 is MultiGain 2.0) to compute the value; the best time is highlighted in bold. For all but one benchmark, Storm outperforms both MultiGain and Pet. The single exception is mer (4), where Pet is slightly faster, leveraging the fact that only a very small part of the model has to be explored.

Data availability. All model files used for the comparison, as well as the resulting log files, are available at DOI 10.5281/zenodo.8219191 [100].

4.3 Outlook

We pinpoint several streams of research currently pursued. First, the algorithms have been extended to **multiple average rewards** [35, 61] and we refer to

Table 4. Performance comparison results of tools for average-reward properties

Model (Parameters)	# states	Value	Storm	MG 2.0	Pet
busyRing	1,912	1.0	**0.04**	1.66	2.42
coin (`N`=2, `K`=5)	656	1.0	**0.01**	0.57	3.36
consensus (`N`=4, `K`=10)	104,576	1.0	**4.83**	189.95	4,505.29
csma (`N`=2, `K`=4)	4,958	1.0	**0.04**	1.52	4.49
cs_nfail	184	0.33	**0.02**	0.48	2.85
eajs (`energy_capacity`=500)	93,228	3.64	**0.36**	11.02	168.80
eajs (`energy_capacity`=1000)	193,728	3.64	**0.71**	36.61	345.28
firewire (`deadline`=20, `delay`=2)	2,862	0.0	**0.04**	1.15	6.22
ij (10)	1,023	1.0	**0.02**	0.83	3.61
investor	6,688	0.95	**0.07**	4.19	5.15
mer (3)	15,622	1.5	**0.81**	12.77	16.17
mer (4)	119,305	1.5	41.82	2,385.99	**41.52**
pacman (`MAXSTEPS`=10)	6,852	0.78	**0.16**	3.41	4.71
pacman (`MAXSTEPS`=15)	96,894	0.99	**2.95**	14.31	9.77
pnueli-zuck	2,701	1.0	**0.04**	1.31	3.91
rabin (3)	15,622	0.86	**0.23**	11.08	8.25
sensors (`K`=5)	267	0.45	**0.01**	0.60	1.97
virus (3)	809	0.0	**0.02**	1.15	2.67
wlan (`COL`=6, `MAX_BACKOFF`=3)	284,446	0.0	**1.26**	9.04	24.82
zeroconf	1,088	0.0	**0.02**	0.70	1.79

Sect. 6 for a discussion of the achievements and challenges. Second, some approaches handle **unknown models** that can only be simulated [2,153], or avoid their construction for efficiency reasons [148]. Finally, while value iteration is the prevailing solution approach, **guarantees on its precision** (stopping criteria) have only been given recently [12,152].

5 Linear Temporal Logic

The traditional analysis of MDPs, in particular in the context of operations research and performance optimization, is based on rewards. In domains such as AI, robotics, or economics, it is often the discounted reward; in other contexts, where the steady state or the long-run behaviour is more relevant, it is the average reward (see Sect. 4). However, in the context of verification, be it of hardware, software or cyber-physical systems, not only reachability but also more complex temporal properties are required [191]. While the analysis of branching-time properties typically boils down to reachability analysis, the analysis of linear-time properties is more complex. The most prominent formalism for capturing linear-time properties is the *linear temporal logic* (LTL) [191].

The standard way to analyse LTL properties is the automata-theoretic approach [213]: The formula is translated to an automaton and, subsequently, the product of the system and this automaton is analysed. While LTL properties occurring in verification of hardware or non-stochastic software tend to be very complex (see e.g. the LTL Store collection [149]), this is less pronounced for stochastic systems. One of the main reasons was the infeasibility of obtaining small automata apt for probabilistic model checking. Indeed, instead of nondeterministic Büchi automata (NBA), for which good translators had long been available, some degree of determinism is needed for MDP. Until recently, Rabin automata produced by determinisation were the default but hardly scalable solution. For instance, about 10 years ago, one fairness contract was translated by the then state-of-the-art methods within Prism to an automaton with 4 states, while a conjunction of two yielded over ten thousand states, and a conjunction of three would not finish computing in a week [147].

In the decade since [147], alternative approaches started flourishing. They avoid the determinisation by direct translations [81] or by employing weaker forms of determinism, such as limit-determinism [108, 205]. The resulting tools such as Rabinizer [151] or spot [77], or libraries such as Owl [150], now reach the same level of scalability as for nondeterministic automata, allowing for verification of more complex properties. Model checkers such as Prism today (i) contain a pre-computed, built-in set of automata for the handful most used properties, and (ii) can link external translators to provide the state-of-the-art-sized automata via the Hanoi Omega-Automata (HOA) standard format [15]. Consequently, comparing the efficiency of model checkers themselves is not very relevant in this category; instead, we describe the main line of model checking algorithms, and discuss the limitations and additional features of the tools.

5.1 Algorithms and Tool Support

The standard algorithm (i) constructs the product of the system and the automaton (with a given acceptance condition), (ii) identifies the maximum accepting end components, and (iii) computes the reachability probability of their union. Step (ii) depends on the particular acceptance condition. The default since the inception of Prism was the Rabin condition due to the better efficiency compared to Muller or parity. Since the appearance of the new translations, the algorithm has been extended to generalized Rabin [56] within Prism and limit-determinism in the Prism-based MoChiBA [206] as well as in a lazy variant [108]. Further improvements on the sizes and types of the automata (e.g. good-for-MDP, Emerson-Lei) followed [82, 110, 135, 174, 181].

Several current tools can be applied to LTL and related specifications:

epmc supports the verification of MDPs against LTL and PCTL*. It translates formulae to an NBA using spot and then applies the lazy approach [108] to compute its satisfaction probability.

MultiGain 2.0 is an extension of Prism capable of verification and strategy synthesis for LTL properties combined with long-run average rewards and steady-state constraints.

Prism itself supports LTL and PCTL* model checking for MDPs. LTL model checking is done via deterministic ω-automata, typically (generalised) Rabin automata, simplified to Büchi or finite automata if appropriate. The translation uses a custom version of `ltl2dstar`, combined with a built-in automata library; alternatively, it can connect to external translators via the HOA format. For the subclass of (co)safe LTL, Prism also supports cumulative expected reward until satisfaction properties.

Storm answers LTL, lexicographic multi-objective LTL [60], as well as PCTL* queries for MDPs. It uses deterministic ω-automata with general Boolean acceptance formulas obtained from spot.

5.2 Outlook

After a decade of research on alternative translations and automata types, the performance both in terms of runtime and of the near-minimality of the size of the automata have reached practical applicability. A few question remain open, such as whether the semantic notion of **good-for-MDP automata** allows us to produce yet smaller automata efficiently, compared to the syntactically defined acceptance conditions. However, the main focus should now move to **modelling and applications**. For LTL formulae, a decent amount of benchmarks is available. Many sets repeatedly used across different papers have been compiled in LTL Store [149]. However due to the earlier scalability problems in probabilistic LTL model checking, the number of probabilistic models coming together with more complex LTL specifications remains quite limited so far [119,164].

6 Multi-Objective Analysis

System performance is commonly assessed with respect to multiple quantities such as the probability of a crash, the expected average energy consumption, or the expected time until task completion. System designers have to consider the interplay between these quantities: minimising the task completion time might require actions that increase the likelihood of a crash. *Multi-objective analysis* [62,83] reveals trade-offs between the considered quantities by showing which compromises are achievable. The system is given as a nondeterministic model $\mathcal{M}$ while the quantities are specified as a vector $\langle \varphi_1, \dots, \varphi_\ell \rangle$ of $\ell \geq 2$ objectives. The objectives commonly refer to rewards attached to states or transitions of $\mathcal{M}$. An ℓ-dimensional point $\vec{p} = \langle p_1, \dots, p_\ell \rangle \in \mathbb{R}^\ell$ is *achievable* if there exists a single policy for $\mathcal{M}$—i.e. a resolution of its nondeterminism—under which for all $i \in \{1, \dots, \ell\}$ the value of objective φ_i is at

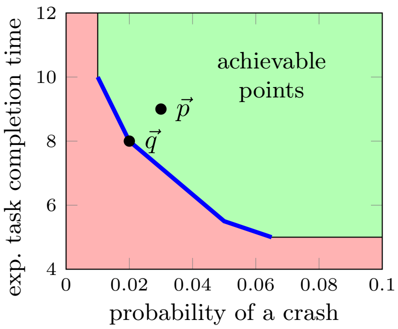


Table 5. Feature comparison of tools for multi-objective verification

	Epmc	Prism	MultiGain	Storm	Prism-games
models	MDP	MDP	MDP	MDP, MA	SG
objectives					
– reach. prob.	yes	yes	no	yes	qualitative
– total reward	yes	yes	no	yes	yes
– LRA reward	no	no	yes	yes	yes
– rew. bounded	no	steps	no	yes	no
– LTL prob.	yes	yes	yes	lexicographic	no
queries					
– achievability	yes	yes	yes	yes	yes
– numerical	yes	yes	yes	yes	no
– Pareto	no	$\ell = 2$	$\ell \leq 3$	yes	$\ell = 2$

least (or at most) p_i. Multi-objective analysis answers queries concerning the (set of) achievable points.

As an example, the green area in the figure above on the right shows the set of achievable points for $\ell = 2$ objectives as labelled on the axes. Point $\vec{p} = \langle 0.03, 9\rangle$ is achievable but *dominated* by other achievable points; $\vec{q} = \langle 0.02, 8\rangle$, for example, yields "better" values for both objectives. The blue line fragments indicate the set of undominated solutions—the *Pareto front*.

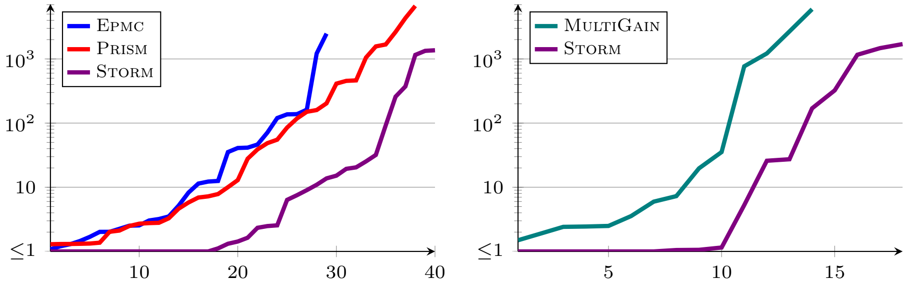
6.1 Algorithms and Tool Support

Table 5 compares quantitative verification tools in terms of their support for multi-objective analysis. We consider the supported kinds of models (where SG are stochastic games), objectives, and analysis queries. For the latter, we follow [88] and distinguish (i) achievability queries, asking whether a given point $\vec{p}$ is achievable, (ii) numerical queries asking for the optimal value for one objective while the others have to achieve a given $(\ell-1)$-dimensional point, and (iii) Pareto queries, asking for (an approximation of) the Pareto front.

We elaborate on the features of the individual tools:

Epmc supports multi-objective achievability and numerical queries for MDPs over total reachability reward objectives as well as objectives specified in LTL. The latter was applied to solve probabilistic preference-based planning problems [170]. Epmc implements the algorithm of [88] based on value iteration.

MultiGain is an extension of Prism that implements the linear programming-based approach of [35] for multiple steady-state and LRA reward objectives on MDPs to answer achievability, numerical, and Pareto queries—the latter for up to $\ell = 3$ objectives. A recent extension MultiGain 2.0 [23] also incorporates the methods of [61, 146] to add support for mixtures of steady-state, LRA reward, and LTL specifications.



(a) Reachability and total reward obj. (b) Steady-state and LRA reward obj.

Fig. 1. Performance comparison results of tools for multi-objective verification

Prism answers achievability, numerical, and Pareto queries for MDPs over combinations of total reward, step-bounded reward, and LTL specification objectives. It implements methods based on value iteration [88] and on linear programming [87]. While the latter only works for achievability and numerical queries, the former can also be used to approximate Pareto fronts over up to $\ell = 2$ objectives. Prism's graphical interface allows the user to conveniently examine the results.

Prism-games implements value iteration over convex sets [27] to analyze multiple total and LRA (ratio) reward objectives as well as almost-sure reachability constraints. Prism-games supports Pareto queries for $\ell = 2$ objectives and achievability queries for arbitrary Boolean combinations of objectives. An extension exists towards lexicographic queries for reachability objectives [59].

Storm handles achievability, numerical, and Pareto queries for MDPs and MA [196]. Storm implements the algorithm of [88] for total reachability reward objectives as well as extensions towards reward-bounded reachability objectives [115] and LRA reward objectives [197]. Furthermore, Storm supports multi-dimensional quantile queries [115], lexicographic LTL specifications [59], and multi-objective analysis under non-randomised policies with limited memory [74].

6.2 Performance Comparision

We empirically compare the performance of the tools for solving achievability queries on MDPs with (i) reachability and total reward objectives as well as (ii) steady-state and LRA reward objectives. We consider various benchmark models and objectives from the literature, e.g. [37, 88, 115, 119, 164]. To obtain challenging achievability queries, the queried points $\vec{p} = \langle p_1, \ldots, p_\ell \rangle$ have been obtained by roughly setting the threshold p_i for the i^{th} objective to 90% of its optimal (single-objective) value. All experiments ran on an Intel Xeon Platinum 8160 CPU with 8 cores and 32 GB of RAM available. We measured the wall-clock runtimes of the tools and aborted executions after 1800 seconds.

Reachability and total reward objectives. Our benchmark set contains 46 concrete queries over reachability probability and total reward objectives from 8 different model families. These queries are supported by Epmc, Prism, and Storm, which all use the approach of [88] based on value iteration as their default method. On 12 queries, the tools reported inconsistent achievability results. We still include these problematic cases in our evaluation since identifying the *correct* solution is not trivial. The tool runtimes in seconds are summarised in Fig. 1 a. In this *quantile plot*, a point $\langle x, y \rangle$ on a line for a tool means that x out of the 46 queries *each* took at most y seconds to complete with this tool. We see that Storm is faster than both Prism and Epmc. The competition among the latter two is tighter, with Prism taking the lead.

Steady-state and LRA reward objectives. We consider 20 queries over steady-state and LRA reward objectives from 6 model families. We solve these queries using MultiGain and Storm. All reported results were consistent for this set of experiments. Fig. 1 b summarizes the runtime comparison (again as a quantile plot with runtimes in seconds). The implementation in Storm using the methods of [197] outperforms the linear-programming based approach of MultiGain.

Data availability. The benchmark models, scripts to reproduce the experiments, and our tool outputs are available at DOI 10.5281/zenodo.8063883 [195].

7 Parametric Markov Models

Classically, probabilistic model checking assumes that the probabilities on the transitions are fixed and precisely known. This assumption is often unrealistic: In various examples, such probabilities are approximations based on expert knowledge. In other applications, these probabilities reflect design decisions that can be freely made. *Parametric Markov models* replace constant probabilities by (polynomial) expressions over a fixed set of parameters X. A parametric Markov model and a valuation of its parameters induces a standard, parameter-free Markov model. The analysis of parametric Markov models was introduced almost 20 years ago [73, 168] while the tool Param brought first tool support more than 10 years ago [107]. By now, the model checkers Storm, Prism and Epmc have support for parametric models.

Over the last decade, there have been various algorithmic advances that answer a variety of *different* queries about a parametric model [132]. The accompanying algorithms have been implemented in various tools and prototypes and all make different assumptions. Furthermore, not every benchmark is well-suited to motivate a particular query. This situation harms further adoption. In the spirit of QComp, we provide a unified and cleaned-up reference implementation on top of the probabilistic model checker Storm, and an annotated benchmark set for various parametric verification queries on parametric DTMCS (pDTMCs) and parametric MDPs (pMDPs).

7.1 Queries and Algorithms

We formulate the key verification tasks for parametric Markov chains. For conciseness, we assume that we are interested in computing the expected reward until reaching a target state, which generalizes computing reachability probabilities. For pMDPs, we assume that we consider the maximal expected reward. The key queries we identify are as follows:

Feasibility: Find parameter values such that the induced expected reward is above a threshold. The state-of-the-art methods rely on guess-and-verify and guess using sampling [64], gradient descent [122], and convex optimisation [67]; the former methods are fastest with few parameters and the latter are fastest with a larger number of parameters. For pMDPs, the quantification order is first over the parameters and then over the schedulers, i.e. the scheduler may depend on the parameter value chosen. This contrasts with *robust* schedulers that do not allow this [10, 219].

Verification: Show that no parameter values exist such that the induced expected reward is above a threshold. This problem is the dual to feasibility queries, but the universal quantification makes it harder to solve. The state-of-the-art approach employs an abstraction-refinement loop [138] using interval Markov chains and combines this with a graph-based analysis to determine monotonicity of the parameters [208].

Solution function computation: Compute a function that maps parameters to the induced expected reward in the corresponding Markov model. While various dedicated methods exist [85, 107, 138], linear equation solving over the field of rational functions performs great overall [138]. Theoretically, a one-step fraction-free method prevents intermediate blowups [22]. For pMDPs, the shapes of these functions are typically prohibitive.

A formal treatment is provided in [136]. Various other queries have been discussed in the literature. They aim to partition the parameter space [138], repair a model with the best parameter values [26], quickly sample parametric Markov models [92], or check whether the derivative is (globally) positive [207]. Others assume a distribution over parameter values [18, 34].

Practicalities. Typically, approaches limit the type of parameter valuations that are considered; graph-preserving valuations require that the underlying graph does not change. While this does not change the complexity of e.g. feasibility [219], it means that the solution function for pDTMCs is a (continuous) rational function and simplifies preprocessing. Likewise, most approaches assume that all pDTMCs are *simple(x)* [136], which (roughly) means that the transition probabilities are given by affine functions that syntactically sum to one. While it is theoretically relevant to allow real-valued parameter valuations, tools typically restrict themselves to rational (or floating-point) number representations.

7.2 Benchmark Collection

We provide a benchmark collection with 12 benchmark families at

github.com/sjunges/parametric-Markov-models

(with the models also archived at DOI 10.5281/zenodo.10646479 [137]). This benchmark collection includes reference invocations for Storm. The collection includes parametric versions of classical benchmarks [107, 119, 164] as well as benchmarks based on the usage of parametric verification in the analysis of hierarchical MDPs [140] and from the sensitivity analysis of Bayesian networks [202].

7.3 Outlook

We believe that the engineering behind many algorithms is still naive and that there is great potential for **algorithmic improvements**. In particular, the verification algorithms lack severely behind in their scalability, and despite being built on top of probabilistic model checking engines, most algorithms have only been implemented for expected rewards and reachability probabilities. More fundamentally, the **synthesis of robust policies in pMDPs** is an open challenge. A next iteration of QComp could also include parametric CTMCs [38], parametric PTAs [118], or structural parameters [9].

8 Partially-Observable MDPs

A major shortcoming of the classic MDP framework is the assumption that decisions can be made based on *complete* state information. In many domains where reasoning about uncertainty is necessary, this assumption is not realistic. For example, information about the current state of an autonomous vehicle is inherently incomplete as it perceives its environment through imperfect sensors.

Partially observable MDPs (POMDPs) extend MDPs with the notion of *partial observability*. Nondeterminism is resolved not based on the complete state, but on the observable information available to the decision procedure. As such, policies for POMDPs are required to be *observation-based*, i.e. decisions are based on the observations and their history. We consider reachability objectives in POMDPs, i.e. we are interested in the minimal or maximal *reachability probability* of certain states in the system or, alternatively, in the minimal and maximal expected total reward until reaching a set of states. In contrast to fully observable MDPs, where optimal policies for such objectives that are memoryless always exist, in POMDPs memory is crucial even for sub-optimal policies.

While POMDPs are widely used for planning in domains like artificial intelligence [201], many verification and synthesis problems have proven to be undecidable. For example, even determining if the reachability probability of a set of states in a POMDP exceeds a threshold is undecidable [172]. Tool support for POMDPs exists in the AI community [78, 155], however, these tools focus on *discounted* objectives, often over a finite horizon [204]. In recent years, efforts took place to extend the tool support for the verification setting of infinite-horizon objectives *without* discounting, also called *indefinite-horizon* objectives. Due to the undecidability of key problems in this setting, the applied methods focus on approximating values and synthesising *good* (sub-optimal) policies with respect to the objectives.

8.1 Algorithms and Tool Support

We showcase three tools from the formal methods community that deal with verification and policy synthesis for POMDPs.

Prism includes support for POMDPs as well as a partially observable variant of PTA. It solves probabilistic reachability or expected cumulative reward queries using the model checking algorithm of [188], which implements a grid-based approach [171, 222] for computing an over-approximation of the objective value on an abstraction of the infinite, fully observable *belief MDP* of the POMDP using only a finite number of beliefs. The resulting policy is then solved to yield an under-approximation which, together, provides lower and upper bounds on the objective value for the POMDP. If the bounds are not tight enough, the approximation can be refined by increasing the grid resolution. The implementation builds upon Prism's Java-based explicit-state engine. The tool then allows the resulting policy to be visualised or simulated in its graphical user interface.

Storm has support for POMDPs that is actively in development. In contrast to Prism, over- and under-approximations can be computed independently of each other. For over-approximations, Storm implements an improved version of the grid-based approach from [171]: it allows for variable grid resolutions for different observations and on-the-fly generation of the belief grid [32]. For under-approximations, Storm uses *belief unfolding with cut-offs*: the belief MDP is unfolded starting in the initial belief. After a fixed number of beliefs has been unfolded, the objective value for all beliefs that have not yet been fully expanded are approximated. This approximation is based on values computed on the POMDP itself using *some* observation-based policy [33]. These values can be computed heuristically by Storm or provided externally. The abstract MDPs are then checked using Storm's MDP model checking core. In addition to the quantitative properties considered here, Storm also supports the verification of *qualitative* properties on POMDPs [139].

Paynt was originally developed for the inductive synthesis of probabilistic programs. In contrast to the model checkers described above, it aims at directly synthesising *finite-state controllers* (FSCs) for POMDPs. An FSC is a Mealy machine that compactly represents a finite-memory policy. To find the best FSC within a given design space of controllers, an MDP abstraction is used, which encodes every possible decision and memory update a policy can make. The resulting process is an over-approximation in the sense that it can simulate every FSC in the design space and switch between FSCs mid-execution. Model checking the MDP yields the best policy within the design space and,

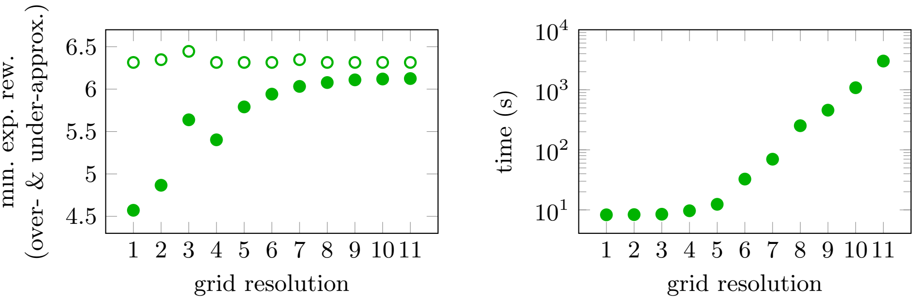


Fig. 2. PRISM's over- and under-approximations for *grid* and computation times

if the policy is not observation-based, a refinement takes place. Additionally, the design space can be pruned by generating counter-examples [8]. For computations on the MDP abstraction and assessing the quality of a synthesised FSC, PAYNT internally uses STORM. As PAYNT synthesises a policy, it can only provide under-approximations of the objective values.

8.2 Performance Comparison

We empirically evaluate the tools described above. As the different approaches and features of the tools make a direct comparison between them misleading, we consider the tools separately on different benchmarks. All three tools accept as input descriptions of POMDPs in the PRISM format. Furthermore, STORM and PAYNT allow inputs in the explicit DRN format.

From the repository of POMDP benchmarks from the literature [8,33,188] at

github.com/moves-rwth/pomdp-collection,

we select one benchmark for each tool to showcase some of its capabilities. All experiments ran on an Intel Xeon Platinum 8160 CPU using 2 threads, 32 GB of RAM, and a time limit of 1 h (measured in wall time). All tools are called using default configurations and options except for the input parameter we evaluate. The short evaluation presented here is not representative of the full capabilities of the tools. Tweaking the configurations used for running the tools typically leads to improvements in the results obtained.

PRISM. We consider the *grid* benchmark, instantiated with length 4 and slipping probability 0.3, for our evaluation of PRISM. The objective here is to minimise an expected reward. We evaluate PRISM with respect to changes in the resolution of the belief grid considered for the over-approximation. Our results are depicted in Fig. 2. The grid-based over-approximation—in the case of minimisation a lower bound—is depicted with solid dots while the under-approximation is depicted with circles. The experiments clearly show the impact of increasing the grid resolution. Generally, the higher the resolution, the better the computed bounds, for the over-approximation as well as the under-approximation which is only

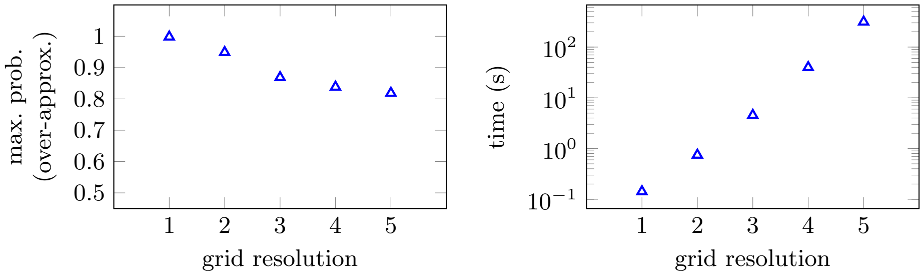


Fig. 3. Storm's over-approximations for *refuel* and computation times

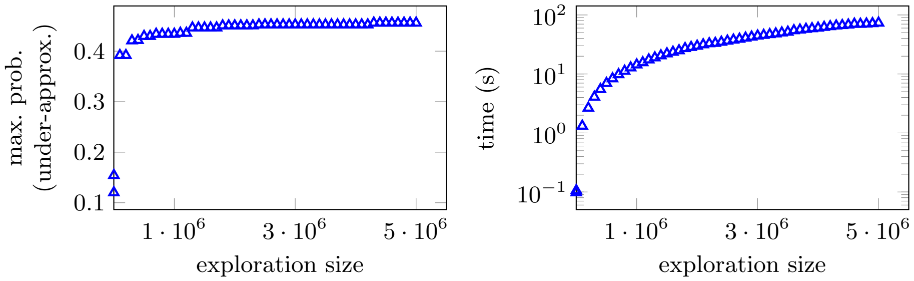


Fig. 4. Storm's under-approximations for *refuel* and computation times

indirectly effected by the grid resolution. However, this improvement comes at the cost of greatly increased runtimes. The outlier at resolution 3 also shows that a good choice of resolution may lead to a tighter approximation of the actual belief space even for rather small resolutions. For resolutions greater than 11, the tool either times out or runs out of memory.

Storm. We evaluate Storm on the *refuel* benchmark instantiated with length 10, computing the maximal probability. We study the over-approximation with respect to changes in the grid resolution and the under-approximation with respect to the size of the unfolding of the belief MDP. Figs. 3 and 4 depict the respective results. Like for the over-approximation in Prism, we observe that an increase in resolution for the grid-based approach leads to tighter bounds on the optimal objective value, with an impact on the runtime. For higher resolutions than 5, Storm ran out of memory on the benchmark. For the under-approximation, we consider unfoldings with increasingly larger state spaces. For larger unfoldings than depicted, we did not achieve better values. We see that with increasing unfolding depth, a tighter bound is achieved. This effect is particularly pronounced in the range of smaller unfolding sizes. With a linear increase in the unfolding size, runtimes appear to scale proportionally.

Paynt. For evaluating Paynt, we select the *grid-avoid* benchmark, instantiated with length 4 and slipping probability 0.1, where the objective is to maximise

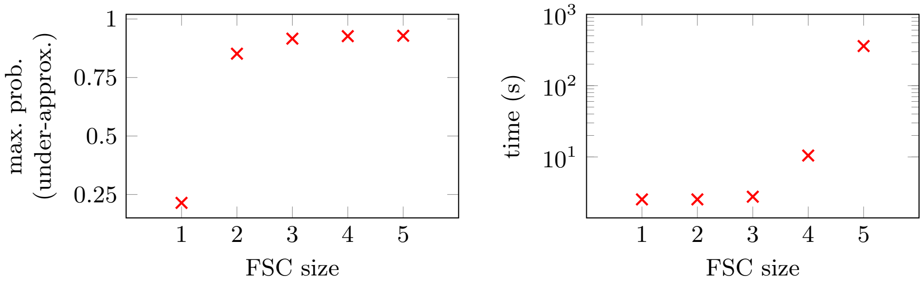


Fig. 5. Max. prob. achieved and time to compute the FSC for *grid-avoid* with Paynt

the probability to reach a target. While a key feature of Paynt is its ability to dynamically increase the size of the considered FSC during computation, we focus on its functionality to compute FSCs of a given size. Thus, we vary the corresponding input parameter. Our results in Fig. 5 show that Paynt is able to obtain FSCs using *some* memory very quickly while FSC quality greatly improves when memory is considered. With increasing FSC size, however, the effect is far less pronounced while runtimes increase drastically. For all FSC sizes greater than 5, the tool timed out in our experiment.

Data availability. The benchmark models used in and the tool outputs generated by our experiments are available at DOI 10.5281/zenodo.8215337 [31].

9 Rare Events

In stochastic systems, *rare events* (RE) stand for measurable events with a positive but very low probability of occurrence. A typical example is the failure probability of highly-reliable systems that can be from 10^{-3} to 10^{-20} or lower.

Formal methods tools can encounter RE in quantitative computations of PRCTL- or CSL-like queries on any model with probabilities. Queries on general stochastic models—i.e. models in continuous-time with residence times or transition firings governed by arbitrary probability distributions—are often estimated via Monte Carlo simulation. This is known as *statistical model checking* and is hindered by RE: since the states of interest are seldom visited, either the number of simulation runs required and thus the runtime grows to impractical values, or the statistical estimations become imprecise or even incorrect. The field of *rare event simulation* (RES) has developed to tackle this problem, and can be divided in two main approaches: *importance sampling* (IS [123]) and *importance splitting* (ISPLIT [141]).

In contrast, exhaustive state-space exploration approaches such as (probabilistic) model checking are not hindered by the rarity of an event, which makes them very attractive to solve RE property queries. However, model checking struggles with the state-space explosion problem, which can be portrayed as the

Table 6. Feature comparison of tools for rare event estimation

Tool		Approach to RE		Models		Semantic formalism
Name	OS	Type	Subtypes	Types	Syntax	
Dftres	Linux, macOS, Windows	RES: IS	Path-ZVA, forcing	DFT, RFT	Jani, Galileo	CTMC, MA (subset)
Fig	Linux	RES: ISPLIT	RESTART-P_j, fixed effort	IOSA, DFT, RFT	IOSA, Jani, Galileo, Kepler	CTMC, IOSA
modes	Linux, macOS, Windows	RES: ISPLIT	RESTART-P_j, fixed effort	any	Modest, Jani	DTMC, CTMC, SHA
Ragtimer	Linux	probabilistic model checking	partial exploration	CRN	Ragtimer	CTMC
St.DftRes	Linux, macOS	RES: ISPLIT	RESTART	DFT	Galileo	CTMC

counterpart of the runtime explosion problem faced by simulation analyses. Additionally, no scalable exhaustive methods are available for models with general distributions. Thus, the challenge here—in the Markovian case—is how to reduce the model size without compromising the correctness or accuracy of the final estimate.

We compare tools that implement RES and model checking to estimate quantitative RE queries in formal stochastic models. Besides defining the scope and capabilities of each tool, we showcase their computation of RE queries in six models with Markovian and arbitrary probability distributions.

9.1 Algorithms and Tool Support

Table 6 summarises the characteristics of tools in formal methods that can estimate rare events. Naturally, the list is not exhaustive: see e.g. earlier works [24, 133, 178]. In QComp 2023, we compare the following tools for rare event scenarios, of which all except Ragtimer are statistical model checkers:

Dftres is designed to estimate transient and steady-state properties of repairable DFTs specified in Galileo, and can also be applied to more general CTMCs and some MAs (that reduce to CTMCs) specified in Jani. It automatically applies IS RES, through the Path-ZVA algorithm [198] and, for transient properties, using forcing [185]. These techniques are particularly applicable for models in which the target event can be reached in a relatively small number of low-probability steps. The algorithms can be applied with no user adjustments, however manual tweaking can improve performance on specific models.

Fig estimates transient and steady state reachability properties in CTMCs and IOSA. It can parse the IOSA and Jani syntax for general models, and Galileo and Kepler for repairable DFTs. Fig automates the use of ISPLIT RES by deriving the importance function from the model and property query [42]. This simplifies the user input to choosing a thresholds-selection algorithm (sequential Monte Carlo or expected success [40]), a simulation run type

(RESTART, RESTART-P_j, or fixed effort), and termination criteria (e.g. by runtime length). There are no theoretical proofs—e.g. of asymptotic efficiency—on the convergence time of the algorithms used by the tool on general models.

modes implements ISPLIT to tackle RES with manually-specified or automatically-derived importance functions much like Fig, including support for the same run types and thresholds-selection algorithms [40, 45].

Ragtimer uses guided stochastic simulation and commutability properties to build a partial state space and acquire a lower-bound for a rare-event probability in chemical reaction networks (CRNs) modeled as CTMCs. It uses reaction information to create a dependency graph, which can demonstrate unreachability. If a property is reachable, it constructs a probability-agnostic model for compositional testing in the IVy tool [177] and uses stochastic simulation to generate a large number of counterexample traces. It then expands these traces and discovers parallel traces by firing commutable reactions and cycles from every state along a trace. The resulting partial state space is passed explicitly to a probabilistic model checker to obtain a lower-bound on the probability of interest. In preliminary testing, Ragtimer finds or approaches the true probability of rare-event properties in several CRN models.

StormDftRes analyses time-bounded reachability properties on (non-repairable) DFTs in either the Galileo or a custom JSON format represented as CTMCs. It aims to perform importance splitting for RE following the ideas of Fig and using the importance functions for DFTs presented in [43].

9.2 Performance Comparison

We demonstrate the capabilities of the tools to compute various PRCTL- and CSL-like property queries on a series of CTMC and SA models. The models used for experimentation are summarised in Table 7: there are six Markovian (CTMCs) and one non-Markovian (SA) models, the latter with hyperexponential and Erlang distributions. All models are provided in the syntax of the tool that specialises in it, and which introduced it to this comparison. They have also been translated to Jani, for the model exchange across tools that enabled this comparison. The SA model is an exception: it is provided in the IOSA and Modest syntaxes alone (for the Fig and modes tools), since it has committed actions that are currently unsupported in Jani.

One or more rare-event properties are given per model: We used the tools to estimate them, showing the results in Table 8. Per model and property we had the tools run for 1, 5, 10, and 30 minutes (indicated in column 🕒) in the TACAS VM [84]. Each tool could use a default run (minimal configuration) or custom commands for that model-property combination. Table 8 reports only one of those values: when the difference between them is below 15% we report the default run; else, we report the one closer to the exact property value [47]. In the table, an empty cell indicates no support for that property/model. Values produced by a custom command are marked with a superscript star *. The values reported are either 95% confidence intervals ($p \pm \delta$), sound lower bounds

Table 7. Models used in the comparison of tools for rare event estimation

Name	Type	Family	Description	Properties
forked-cycle-tandem-queue	CTMC	queueing system	*three queues*: arrivals to Q1; probabilistic output to Q1, Q2, Q3; study overflows in Q2. *(previously unpublished)*	φ_1: $\texttt{P}_{=?}[\,\texttt{q2} > \texttt{0 U q2} \geqslant \texttt{6}\,]$ φ_2: $\texttt{P}_{=?}[\,\texttt{q2} > \texttt{0 U}_{\leqslant 555}\ \texttt{q2} \geqslant \texttt{6}\,]$ φ_3: $\texttt{P}_{=?}[\,\texttt{F}_{\leqslant 555}\ (\texttt{q2} \geqslant \texttt{6})\,]$ $\tilde{\varphi}_4$: $\texttt{S}_{=?}[\,\texttt{q2} \geqslant \texttt{6}\,]$
7nodes-network	SA	queueing system	*non-Jackson 7 queues*: arrivals to all queues; near-full probabilistic interconnection; study overflow in Q7. [215]	$\tilde{\varphi}_5$: $\texttt{S}_{=?}[\,\texttt{n7} \geqslant \texttt{30}\,]$
2react	CTMC	chemical reaction network	*single species production-degradation*: simple 2-reaction system with one shortest trace. [69]	φ_6: $\texttt{P}_{=?}[\,\texttt{F}_{\leqslant 100}\ (\texttt{s2} \geqslant \texttt{80})\,]$
6react	CTMC	chemical reaction network	*enzymatic futile cycle*: 6-reaction system with large state space, cyclic behavior, and one shortest trace. [157]	φ_7: $\texttt{P}_{=?}[\,\texttt{F}_{\leqslant 100}\ (\texttt{s5} = \texttt{25})\,]$
8react	CTMC	chemical reaction network	*modified yeast polarization*: concurrent 8-reaction system with cyclic behavior and many shortest traces. [76]	φ_8: $\texttt{P}_{=?}[\,\texttt{F}_{\leqslant 20}\ (\texttt{G\_bg} \geqslant \texttt{50})\,]$
HECS	CTMC	dynamic fault tree	*hypothetical example computer system*: standard DFT benchmark. [216]	φ_9 : $\texttt{P}_{=?}[\,\texttt{F}_{\leqslant 1}\ \texttt{TLE}\,]$ "*unreliability @ 1*"
MAS	CTMC	dynamic fault tree	*mission avionics system*: highly redundant safety-critical system with hard- and software components. [189]	φ_{10}: $\texttt{P}_{=?}[\,\texttt{F}_{\leqslant 1}\ \texttt{TLE}\,]$ "*unreliability @ 1*"

($\geqslant p$), failures ($\varnothing$), or omitted computations ("). The latter applies to e.g. model checkers like Ragtimer that use one runtime since longer runs are seldom beneficial. In general, smaller confidence intervals and results closer to the true value (indicated in the second column under heading "Prop." as either a statistical approximation $\approx p$ or truncated exact value p, obtained from reference material or computed with higher resources, viz. more memory, runtime, and CPU power) are better.

We note that the default ("sound") run of Dftres can run longer or shorter than the hard time limit, and its renewal-theory implementation cannot compute $\tilde{\varphi}_4$ on that Jani model; also, Fig and modes used crude Monte Carlo (not RES) to analyse the DFTs because no useful importance function could be derived when the dominant failures have short traces; and Ragtimer used one runtime per property, since longer runs are seldom beneficial for model checkers.

Data availability. We provide an artifact allowing a full experimental reproduction at DOI 10.6084/m9.figshare.23818395 [47].

9.3 Outlook

We see the need for further research to unify rare event approaches in the formal tools community, e.g. to allow **automatic identification of the algorithm to use**. A concrete example is the high performance of Dftres (using IS) to analyse the DFTs in contrast to its comparatively low performance for properties

Table 8. Performance comparison results of tools for rare event estimation

Prop.		🕒	Dftres	Fig	modes	Ragtimer	StormDftRes
φ_1	9.23E-10	1	9.2E-10 ± 3E-13	9.0E-10 ± 9E-11*	9.4E-10 ± 6E-11*		
		5	9.2E-10 ± 6E-14	9.5E-10 ± 4E-11*	9.2E-10 ± 4E-11*		
		10	9.2E-10 ± 4E-14	9.0E-10 ± 3E-11*	9.2E-10 ± 2E-11*		
		30	9.2E-10 ± 2E-14	9.3E-10 ± 2E-11*	9.1E-10 ± 8E-11*		
φ_2	9.23E-10	1	9.2E-10 ± 5E-13	9.4E-10 ± 1E-10*	9.3E-10 ± 3E-10*		
		5	9.2E-10 ± 8E-14	9.2E-10 ± 5E-11*	9.1E-10 ± 3E-11*		
		10	9.2E-10 ± 6E-14	9.3E-10 ± 3E-11*	9.2E-10 ± 2E-11*		
		30	9.2E-10 ± 3E-14	9.2E-10 ± 2E-11*	9.3E-10 ± 1E-11*		
φ_3	9.00E-08	1	9.4E-09 ± 3E-09*	8.7E-08 ± 7E-08*	8.1E-08 ± 2E-08*		
		5	8.6E-09 ± 1E-09	7.6E-08 ± 3E-08*	9.2E-08 ± 6E-09*		
		10	1.1E-08 ± 4E-09	8.3E-08 ± 3E-08*	9.0E-08 ± 4E-09*		
		30	1.3E-08 ± 5E-09	9.1E-08 ± 2E-08*	9.1E-08 ± 3E-09*		
$\tilde{\varphi}_4$	5.64E-11	1	∅	6.2E-11 ± 2E-11*			
		5	∅	6.0E-11 ± 6E-12*			
		10	∅	5.8E-11 ± 4E-12*			
		30	∅	5.5E-11 ± 2E-12*			
$\tilde{\varphi}_5$	≈ 7.57E-15	1		7.0E-15 ± 5E-15*	7.1E-15 ± 2E-15*		
		5		8.3E-15 ± 4E-15	7.7E-15 ± 1E-15*		
		10		7.2E-15 ± 2E-15	7.7E-15 ± 6E-16*		
		30		8.8E-15 ± 3E-15	8.3E-15 ± 4E-16*		
φ_6	3.06E-07	1			2.9E-07 ± 1E-08*	⩾ 3.0E-07	
		5			3.0E-07 ± 7E-09*	"	
		10			3.0E-07 ± 5E-09*	"	
		30			3.0E-07 ± 3E-09*	"	
φ_7	1.70E-07	1			1.7E-07 ± 4E-08*	⩾ 2.8E-18*	
		5			1.8E-07 ± 2E-08*	"	
		10			1.7E-07 ± 1E-08*	"	
		30			1.8E-07 ± 8E-09*	"	
φ_8	≈ 1.20E-06	1			1.5E-06 ± 3E-07*	∅	
		5			1.5E-06 ± 1E-07*	⩾ 2.3E-28	
		10			1.6E-06 ± 9E-08*	"	
		30			1.7E-06 ± 5E-08*	"	
φ_9	2.20E-04	1	2.2E-04 ± 6E-06	2.3E-04 ± 3E-05	2.1E-04 ± 3E-05		2.2E-04 ± 2E-05
		5	2.2E-04 ± 5E-07	2.2E-04 ± 1E-05	2.2E-04 ± 1E-05		2.2E-04 ± 7E-06
		10	2.2E-04 ± 2E-07	2.2E-04 ± 1E-05	2.2E-04 ± 1E-05		2.2E-04 ± 5E-06
		30	2.2E-04 ± 1E-07	2.2E-04 ± 5E-06	2.2E-04 ± 5E-06		2.2E-04 ± 3E-06
φ_{10}	1.00E-05	1	1.1E-05 ± 8E-06	8.1E-06 ± 1E-05	6.7E-06 ± 3E-06		1.4E-05 ± 5E-06
		5	1.0E-05 ± 2E-06*	7.3E-06 ± 5E-06	1.2E-05 ± 2E-06		1.0E-05 ± 2E-06
		10	9.7E-06 ± 2E-06	1.0E-05 ± 5E-06	1.1E-05 ± 1E-06		9.8E-05 ± 1E-06
		30	1.0E-05 ± 1E-06	8.9E-06 ± 2E-06	9.7E-06 ± 7E-07		1.1E-05 ± 8E-07

in queueing systems. This in contrast to Fig and modes, which (using ISPLIT) performed well for the latter, but found crude Monte Carlo to be their best approach for the DFTs.

10 Robust Decision-Making Under Uncertainty

In recent years, there has been a strong push to combine the areas of formal verification—in particular model checking—and artificial intelligence (AI). A specific area that is native to both of those areas is *decision-making under uncertainty* [143]. The level and type of uncertainty affect the capabilities of AI systems to make intelligent decisions. The core problem is to provide a guarantee that an AI system, operating under uncertainty, adheres to some formally specified constraint, e.g. given as a temporal logic specification (see Sect. 5). State-of-the-art approaches use models, in particular MDPs, to capture sequential decision-making problems for agents operating in uncertain environments. Moreover, sensor limitations may lead to partial observability of the system's current state, giving rise to POMDPs (see Sect. 8). MDPs augmented with a model of adversarial behaviour are stochastic games (SGs, see Sect. 12) and their partially observable counterpart is a partially-observable SG (POSG).

The likelihood of uncertain events, such as a message loss in communication channels or specific responses by human operators, may only be an estimate from data. The models mentioned above capture uncertainty in the form of precise probabilities—either in their transition dynamics or in their observation models. However, such *point estimates* of probabilities from data carry the risk of statistical errors. Moreover, the optimal policies for agents are usually highly sensitive to small perturbations in transition probabilities, leading to suboptimal outcomes such as a deterioration in performance [96,175].

Uncertainty models, sometimes also referred to as *robust models*, remove this assumption by incorporating uncertainty sets of probabilities. In the literature, uncertain MDPs use, for example, *probability intervals* or *likelihood functions* [187,218]. Similar extensions exist for uncertain POMDPs, where uncertainty may also affect the observation model [39,50,68,130,209]. To the best of our knowledge, there are no results on uncertain POSGs. Fig. 6 shows a family of *uncertainty models*, capturing different types of uncertainty and their relation to each other. The three different types of arrows indicate the addition of (1) adversarial behaviour, (2) uncertainty on probability distributions, and (3) partial observability from one model to another. In the figure, adversarial behaviour increases from left to right. The left and right columns are partially observable models. Finally, the bottom row shows models that (in addition to probabilistic and adversarial behaviour) account for uncertainty in probability distributions. For an overview, we refer the interested reader to e.g. [17].

10.1 Algorithms and Tool Support

Prism [163], a widely-used probabilistic model checker, provides support for two common classes of uncertainty models: *interval discrete-time Markov chains*

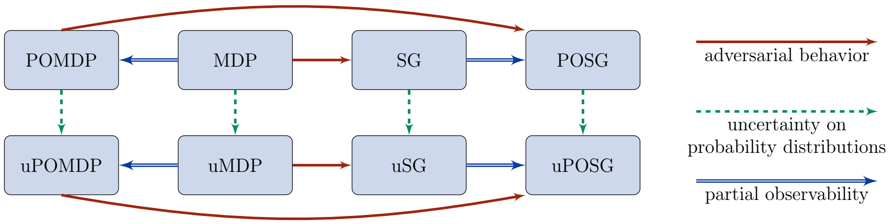


Fig. 6. A family of closely related uncertainty models

(IDTMCs) and *interval Markov decision processes* (IMDPs). These are specified to the tool with a simple extension of the PRISM modelling language where the probabilities attached to variable updates within a guarded command are optionally provided as intervals, for example

$$[move]\ loc = 1\ \rightarrow\ [0.85, 0.95] : (loc' = 2) + [0.05, 0.15] : (loc' = 1);$$[21]

and, as usual, can be given as expressions in terms of variables and parameters:

$$[send]\ s{=}1\ \rightarrow\ [p_{fail} - \varepsilon, p_{fail} + \varepsilon] : (s'{=}1) + [(1{-}p_{fail}) - \varepsilon, (1{-}p_{fail}) + \varepsilon] : (s'{=}2);$$

This makes it straightforward to adapt existing DTMC or MDP benchmarks [164] to their interval variants, as done for example in [193].

PRISM provides *robust* verification, quantifying over all possible transition probabilities contained within the models' uncertainty sets. Property specification extends the existing PRISM property language. For IDTMCs and IMDPs, the tool supports the temporal logic PCTL, extended with (expected) reward operators and (co-safe) LTL formulae. For example, formulas $\mathtt{P}_{\mathrm{maxmin}=?}[\,\mathtt{F}\ goal\,]$ and $\mathtt{P}_{\mathrm{maxmax}=?}[\,\mathtt{F}\ goal\,]$ ask for the worst- and best-case scenarios, respectively, for maximising the probability of reaching a *goal*-labelled state.

Like many probabilistic model checking implementations, the uncertain models are solved via dynamic programming, in this case, *robust value iteration* [187, 220], implemented in PRISM's Java-based explicit-state model checking engine. Optimal policies for IMDPs can be generated and exported or simulated. Access to IDTMC and IMDP model checking is also provided programmatically at an API level, and has been applied to various problems, including anytime model learning [210] and abstraction of dynamical systems [19].

10.2 Outlook

Tool support for uncertainty models can be extended in various directions, for example to provide model checking for some of the model classes identified in Fig. 6 featuring partial observability (uncertain POMDPs) or adversarial behaviour (uncertain SGs), as well as improving efficiency and scalability for the

[21] In this example, *move* labels the transitions induced by the command, $loc = 1$ is the guard that determines when the command is enabled, and each of the two branches to the right of → has an interval of probabilities and a set of assignments.

simpler model classes. It will also be beneficial to extend the range of uncertainty types beyond intervals, which also necessitates more significant modelling language extensions.

11 State Space Exploration

State space exploration engines form the foundation of numerous quantitative analysis tools, playing a pivotal role in their functionality. Explicit-state model checkers, such as Storm with its `sparse` engine and mcsta, rely on exploration engines to exhaustively construct the complete state space of a model before applying probabilistic model checking algorithms. Additionally, statistical model checkers such as modes leverage exploration engines to generate large amounts of traces for statistical analysis. Exploration engines have recently also been used for training and verifying reinforcement learning agents [97, 99].

In an effort to better understand the performance characteristics of the exploration engines utilised in different tools, we systematically benchmark and compare them. For this purpose, we consider the time and space needed for building an explicit representation of the complete state space of a model. Additionally, we compare the engines based on qualitative criteria such as the types of models they can handle and the interfaces they provide.

11.1 Tool Support

The tools participating in this category are the Modest Toolset, Momba, and Storm. Both Momba and Storm participate with multiple engines, adding further diversity to the evaluation. Since all three tools support Jani, we employ it as a foundation for comparing and contrasting their capabilities.

The Modest Toolset includes a state space exploration engine written in C# that is used by several of its tools, including mcsta and modes. It supports all types of models specified by Jani, including all Jani extensions. In that regard, it stands out as the most versatile among the engines we consider. For PTA, the engine supports the digital clock semantics [165], explicit valuations, clock regions [120], as well as clock zones [70]. In addition, it supports a symbolic treatment of continuous variables for hybrid models. In contrast to both Storm and Momba, which both provide public interfaces to their engines, the Modest Toolset's engine is intended for internal use only and does not provide a public interface. The Modest Toolset includes a separate mopy transpilation tool to convert models to Python code implementing a first-state-next-state interface which can be used to explore the model's state space. In our experiments below, we access the Modest Toolset's state space exploration engine via mcsta.

Momba includes as a key feature a state space exploration engine designed to make exploration readily accessible via a comprehensive Python API. To achieve good performance, the engine is written in Rust. While Momba itself supports all of JANI, its state space exploration engine is more limited: It supports all discrete-time model types and flavours of timed automata specified by JANI except stochastic timed automata. The supported JANI extensions are `arrays`, `derived-operators`, `named-expressions`, and `trigonometric-functions`. In particular, the `functions` extension is not supported yet. For timed automata, it supports explicit valuations as well as clock zones. The Python API also provides functionality that goes beyond mere exploration: for instance, arbitrary JANI expressions can be evaluated in a given state and, for timed automata, clock zones can be manipulated. In addition to its traditional state space exploration engine, MOMBA also participates with an experimental new engine supporting a parallelized exploration mode harnessing the potential of multi-core systems. This experimental engine does not currently support timed automata and is not yet exposed via the Python API.

STORM participates with its `sparse` and `dd-to-sparse` engines. While STORM's `sparse` engine, like the engines of the MODEST TOOLSET and MOMBA, adopts a conventional explicit-state approach, the `dd-to-sparse` engine is based on first constructing a symbolic representation using BDDs of the state space and subsequently translating this to a traditional explicit representation. STORM supports all discrete- and continuous-time model types specified by JANI, except timed and hybrid automata. The supported JANI extensions are `arrays`, `derived-operators`, `functions`, and `state-exit-rewards`. STORM provides both a C++ and a Python interface, the latter as part of STORMPY, to its state space exploration engine. While the C++ API is fully featured, the Python API only supports the exploration of the entire state space of JANI models (but not the simulation of individual traces) while it has no such limitation for PRISM models. In contrast to Modest Toolset and MOMBA, STORM offers support for arbitrary-precision arithmetic using rational numbers implemented in the GMP library. This enables precise calculations and analysis, particularly when dealing with models that require high precision.

11.2 Performance Comparison

In our experimental evaluation, we utilise the QVBS as the foundation for benchmarking the tools. To ensure a meaningful comparison, we focus exclusively on discrete-time models, as these are supported by all the participating tools. Out of our initial 229 QVBS benchmarks, 25 resulted in timeouts after 30 minutes or were unsupported by all tools. Hence, the following analysis focuses on the remaining 204 benchmarks. For each benchmark, we measure the time required by each state space exploration engine to construct the entire state space. Additionally, we track the number of states counted by the engines and assess the memory consumption associated with each state where applicable. All benchmarks ran on a computer equipped with a 16-core AMD EPYC-Milan processor running at 3.4 GHz and 128 GB of RAM.

Table 9 shows the number of benchmarks per tool and our experiments' qualitative outcomes: we display the number of benchmarks that were successfully

Table 9. Number of benchmarks per outcome and state space exploration engine

Engine	*solved*	*unsupported*	*timeout*	*error*
Modest Toolset	194	9	1	0
Momba (v1)	159	45	0	0
Momba (v2,`seq`)	159	45	0	0
Momba (v2,`par`)	154	45	5	0
Storm (`dd-to-sparse`)	195	3	2	4
Storm (`sparse`)	202	0	2	0

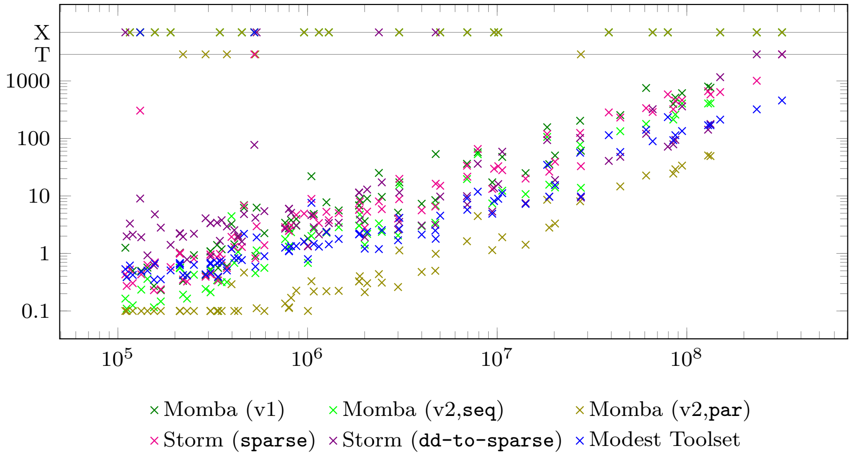


Fig. 7. Runtimes in seconds in relation to the total number of states

solved, *unsupported*, lead to a *timeout*, or caused an *error*. The 9 benchmarks not supported by the Modest Toolset's engine use a complex specification for the initial states. The 45 benchmarks not supported by Momba use the `functions` Jani extension and are a superset of the 9 benchmarks not supported by the Modest Toolset. The 3 benchmarks not supported by Storm's `dd-to-sparse` engine use assignment indices while for 4 benchmarks the same engine returned an error due to the BDD implementation running out of memory. The timeouts are all for different benchmarks. While the number of states reported by Storm and Momba is the same for all benchmarks and engines, the Modest Toolset sometimes reports fewer states which presumably is due to some state space-reducing optimizations.

Runtimes. Fig. 7 depicts the running time for each benchmark (on the vertical axis) in relation to the total number of states of the respective benchmark (on the horizontal axis). The marks at the top indicate timeouts (T), and unsupported benchmarks as well as benchmarks returning an error (X). The quantile plot in Fig. 8 presents the cumulative number of benchmarks solved within a

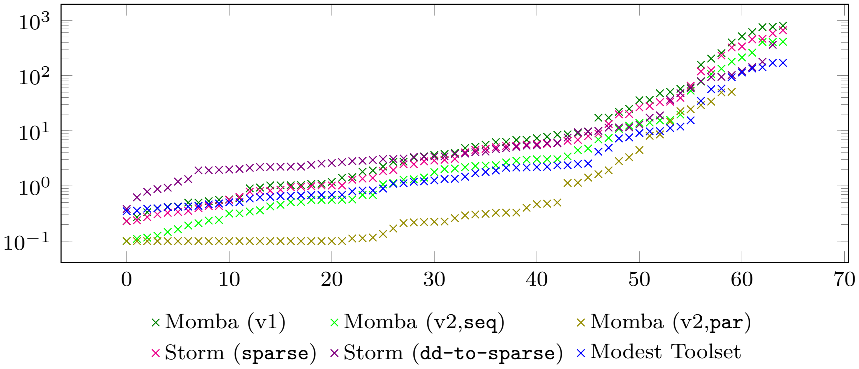


Fig. 8. Runtimes (s) vs. the number of benchmarks each solved in that time

certain time. For presentation purposes, we chose to clamp the running times at $0.1\,s$ and restrict the plots to benchmarks with more than 10^5 states. For smaller benchmarks, the differences in runtimes are practically insignificant. Additionally, Fig. 8 is restricted to benchmarks supported by all engines to prevent skewing the plot (as otherwise an unsupported benchmark and a timeout would have the same effect).

From these results, it is evident that the approach taken by the `dd-to-sparse` engine of Storm only pays off for larger models; even then, it is rarely faster than the conventional explicit engine of the Modest Toolset. Among those engines exclusively using a single core, the Modest Toolset engine is almost always the fastest, although it has a larger startup overhead. This does not come as a surprise because, for efficiency, it is based on compiling Jani models to C# bytecode that is JIT-compiled. Storm's `dd-to-sparse` engine, like Momba's experimental parallel engine (v2,`par`), uses multiple cores since the underlying BDD implementation in Sylvan [75] is parallelised. Momba's parallel engine is always faster than any other engine for benchmarks of a significant size. The average speed-up when compared to its sequential version is a factor of 9.1. In general, though, the runtimes of all engines are often quite similar.

Note that, as Storm and mcsta are model checkers, they do a bit more work than Momba by creating a sparse matrix representation of the transitions and computing atomic propositions. We expect the performance impact of this to be minor—however we did not measure it.

Memory consumption. Another interesting dimension when it comes to state space construction is the required memory. Efficiency is crucial here given the often huge state spaces due to the state space explosion problem. For the traditional explicit state engines, the size of the state space is linear in the number of states. Fig. 9 shows the size of the state spaces in relation to the number of states, computed based on the number of states and the size of each state. Note that the sequential and parallel variant of Momba's experimental engine use the same

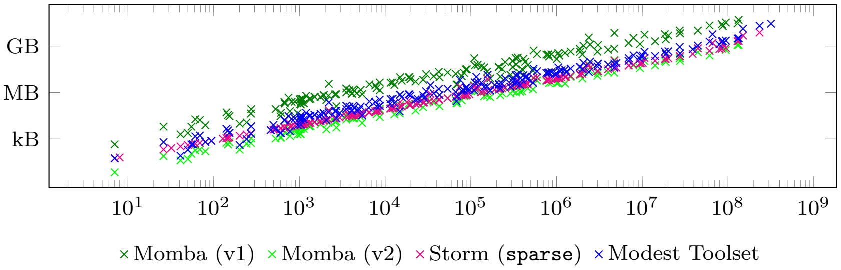


Fig. 9. Size of the state space in relation to the number of states

representation. In contrast to the Modest Toolset, Storm's `sparse` engine and Momba's experimental engine use a more space efficient bit-packing representation of states, thereby reducing the amount of required memory. Momba's original engine uses the worst representation and always requires at least 16 bytes per variable independent of its actual domain.

Summary. Our results show that all engines are roughly comparable with respect to the time it takes to construct the entire state space of a model. Storm's `dd-to-sparse` engine may only be advantageous in terms of runtime for some large models while incurring a high overhead for small models. Among single-core engines, the Modest Toolset's engine is almost always the fastest, especially for large models, while being the most versatile at the same time. The experimental parallel engine of Momba demonstrates that parallel state space exploration can be highly beneficial for larger models. The original Momba engine requires significantly more memory than all others. The Modest Toolset's engine, however, does not provide a public API. Thus, if integration into another tool is a concern, Storm and, in particular, Momba with its original engine have an advantage as they both provide a Python API in addition to APIs in C++ and Rust, respectively.

Limitations. One of the motivations of this category is the lack assessment for simulation of individual traces. Note that the performance characteristics displayed here may not carry over to simulation of individual traces as there is a difference between always computing all successor states, as required for exhaustive exploration, and selectively computing only individual successor states which is, for instance, explicitly supported by Momba. Additionally, an exhaustive exploration requires maintaining a (hash) set of all visited states.

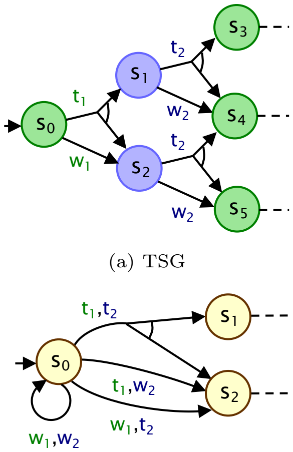
Data availability. An artifact allowing to reproduce the performance comparison is archived and available at DOI 10.5281/zenodo.10626177 [144].

12 Stochastic Games

Game theory provides an effective way to model strategic interactions between multiple agents (or players) collaborating or competing to achieve objectives. Games have long been of interest within formal verification, providing a natural way to model, for example, honest and malicious participants in a security protocol or a controller operating in an adversarial environment.

In the context of quantitative verification, *stochastic games* (SGs) are a natural model to reason about strategic interactions in the context of uncertainty, noise, or randomisation. Verification problems for SGs have been studied for over 20 years [57]; the first model checking tools for SGs appeared over 10 years ago [63], and there has been growing interest in the topic recently.

In essence, SGs (visualised on the right) generalise MDPs by permitting multiple players to have distinct strategies about how to resolve nondeterministic choices in the model. The simplest model, a turn-based SG (TSG), simply partitions the state space of an MDP, with the choices in each state being under the control of exactly one player. A concurrent SG (CSG) provides a more realistic model of concurrent decision-making: in each state, players resolve their choices independently.



(a) TSG

(b) CSG

Verification of SGs also takes a variety of flavours. The simplest option is a *zero-sum* setting, where one player (or a coalition of players) aims to maximise some objective, such as the probability of reaching a set of target states or satisfying a temporal logic formula, and the other player (or players) have the opposite objective, i.e. to minimise it. For SGs, the logic rPATL [63] is widely used, which generalises the well-known game logic ATL [5] to a variety of quantitative objectives. Beyond zero-sum properties, temporal logics and model checking algorithms have been extended [159] to support *equilibria*, which are joint strategies where each player optimises their own distinct objective in such a way that it is not beneficial for any player to unilaterally change strategy.

12.1 Algorithms and Tool Support

Despite verification problems for SGs typically having a higher complexity than their MDP counterparts, core properties of TSGs can be effectively analysed with similar methods such as value iteration [66] or interval iteration and its variants [13, 80]. Methods to solve CSGs tend to be more expensive: again they are usually based on value iteration, but require the solution of a linear programming or equilibrium synthesis problem [159] for every state at each iteration.

Verification tools for SGs under active development are Prism-games and its extensions, Tempest, Pet, and Epmc. We provide a brief empirical comparison of the first four below. These tools share a common input format for SGs, namely the Prism-games modelling language. This extends the widely used Prism modelling language: In the case of TSGs, it is a rather simple extension of the case for MDPs, defining a set of players and the states they own; CSGs use a different model of parallel composition and additional language features.

Epmc also supports the analysis of stochastic parity games and verification of epistemic properties on probabilistic multi-agent systems in addition to its standard probabilistic model checking functionality.

Pet has recently been extended to support reachability objectives for TSGs. It uses Prism-games to parse and explore games, and employs the interval iteration approach of [80] to solve them. Implementing partial exploration based on [80, Sect. 5] in combination with the approach of [152] for more complex objectives such as total reward or mean payoff is planned.

Prism-games mainly focuses on TSGs and CSGs, but it also supports turn-based probabilistic timed games. The tool supports a wide range of zero-sum properties (probabilistic reachability, expected rewards, co-safe linear temporal logic and multi-objective specifications) as well as (social welfare) Nash equilibria. Recent extensions add support for correlated equilibria and social fairness [160]. The implementation is primarily based on variants of value iteration, implemented in Java with explicit state data structures, but also includes symbolic (MTBDD-based) model checking of TSGs [161].

Tempest extends Storm to TSGs with a focus on synthesizing most-permissive strategies. The tool supports zero-sum properties, namely probabilistic reachability and mean-payoff properties. The model checking procedures are based on variants of value iteration using explicit representations of the state space.

[154] and [13] present an extension of Prism-games which adds various methods for solving TSGs: interval iteration (II) [80] and optimistic value iteration (OVI) [13], as well as topological variants of each; the "widest path" (WP) variant of II [190]; and solution methods based on strategy iteration and quadratic programming. The latter are omitted from our comparison since they are fundamentally different from the variants of value iteration employed by the other tools [154, Sect. 5.5.3]; we refer to [154, Sect. 5] for a practical comparison of these solution methods.

Also relevant are Gist [58] and GAVS+ [65], which implement TSG verification, but are no longer developed or maintained, and Uppaal Stratego [72], which supports stochastic priced timed games via multiple other Uppaal branches.

12.2 Performance Comparison

We give a brief performance comparison of the various tools and techniques, focusing on the problem class supported by all tools: zero-sum probabilistic reachability for TSGs. Table 10 shows total runtimes (game construction and solution) on an indicative set of benchmarks from the PRISM Benchmark Suite [164] and [154]. Experiments ran on an AMD Ryzen 5 3600 system, pinned to a single

Table 10. Performance comparison results of tools for stochastic games

Benchmark			Value iteration (s)			ε-exact (s)			
Model + property [parameters]	Param. values	# states	Prism-games (expl.)	Prism-games (symb.)	Temp-est	Pet	P-g+ (II)	P-g+ (OVI)	P-g+ (WP)
avoid + *find* [X_MAX, Y_MAX]	10, 10	106,524	16.9	15.4	**1.4**	**5.0**	17.2	22.4	16.7
	15, 15	480,464	125.9	62.6	**4.7**	**15.7**	126.9	137.2	126
	20, 20	1,436,404	*T/O*	240.8	**12.9**	**57.5**	*T/O*	*T/O*	*T/O*
hallway_human + *save* [X_MAX, Y_MAX]	5, 5	25,000	2.5	1.8	**0.9**	2.9	**2.4**	**2.4**	**2.4**
	10, 10	400,000	10.5	**2.0**	12.9	**9.5**	11.3	11.2	11.3
	15, 15	2,025,000	50.1	**4.0**	101.3	**39.6**	57.0	55.4	56.6
investors + *greater* [N, vmax]	2, 20	568,790	21.8	7.4	**4.9**	**16.7**	33.2	42.3	54.6
	2, 40	2,041,690	98.8	26.0	**19.8**	**69.0**	144.8	183.2	314.6
	3, 20	4,058,751	167.7	**19.2**	39.7	**152.6**	241.4	321.3	484.8
	3, 40	14,569,251	*M/O*	**62.8**	171.2	*T/O*	*T/O*	*T/O*	*T/O*
safe_nav + *reach* [N, feat]	8, D	2,592,845	544.2	**16.2**	518.5	519.4	498.4	508.7	**485.7**
	8, A	17,052,941	*T/O*	**110.8**	*T/O*	*T/O*	*T/O*	*T/O*	*T/O*
BigMec + *BigMec* [N]	10,000	20,003	46.9	9.3	**2.5**	**17.6**	*T/O*	49.5	73.5
	25,000	50,003	290.4	45.8	**12.7**	**82.9**	*T/O*	294.2	472.4
ManyMec + *ManyMec* [N]	10,000	30,002	160.7	263.3	**16.7**	**104.3**	*T/O*	*T/O*	460.4
	25,000	75,002	*T/O*	*T/O*	**98.6**	*T/O*	*T/O*	*T/O*	*T/O*

core and restricted to 8 GB of RAM, running inside Docker, using OpenJDK JRE-17 for all Java tools, and with a timeout (T/O) of 10 minutes. For each invocation, a fresh docker container is created.

For a fair comparison, we group them into two distinct categories based on the degree of accuracy provided: "value iteration" (i.e. no strict guarantees on the correctness of the result) and "ε-exact" (the result is guaranteed to be within $\pm\,\varepsilon = 10^{-6}$ of the true value), marking the fastest tool in each category in bold.

Value iteration. Comparing explicit-state implementations, Tempest is faster than Prism-games on almost all instances (primarily, it appears, due to the former being implemented in C++, but the latter also uses slower but more extensive precomputations). Prism-games, in symbolic mode, outperforms Tempest on most larger models and scales to the biggest TSGs of all tools. Symbolic model building times (not shown) are also usually faster.

ε-exact. Pet outperforms the approaches in the Prism-games extension of [13,154] (denoted P-g+ in the table) on practically all models. This is interesting since the algorithmic approach in the former is the same as interval iteration (II) in the latter. Since these tools are implemented in the same language (Java) and use the same model construction (Prism's model generator), the (significant) differences are solely a result of engineering. Times for the methods in the Prism-games extension are typically in the same order of magnitude, however there are models where one approach significantly outperforms all others.

Data availability. All tools, models and scripts needed to replicate our results can be found at DOI 10.5281/zenodo.7831387 [180].

12.3 Outlook

Interest has grown in the formal verification of SGs in recent years and it has already been applied to a range of domains, from computer security to adaptive software architectures (as evidenced by the collection of Prism-games case studies at prismmodelchecker.org/games/casestudies.php). In addition to improving the efficiency and scalability of existing tools, one key challenge is to develop methods for **partially observable** variants of SG models. Another is to develop support for **richer specification languages**, for example incorporating strategies, equilibria or epistemic properties.

13 Conclusion

We have described the state of the art in tools and algorithms at the frontiers of quantitative verification in ten different categories, covering 19 different tools. In several categories, we reported on the first systematic performance comparison among the included tools. On many of the frontiers we described, tool support for advanced properties and models is now being consolidated, but a plethora of open questions and unimplemented ideas remain for future work. We hope that this report can serve as an inspiration for further work on quantitative verification tooling, and that several of QComp 2023's categories can evolve into regular, serious performance evaluations among competing tools in the near future. At the same time, it is clear that our coverage of the quantitative verification frontiers is not complete. As one example, we mention the area of parametric models based on timed automata (in which parameters are traditionally more structural in nature than the ones in the parametric Markov models of Sect. 7) where tools are maturing [6] and benchmark sets with support for Jani are being collected [7], laying the foundations for future performance evaluations.

For the next edition of QComp, which at the latest will take place in time for the next edition of the TOOLympics, we intend to keep the multiple-category setup. We plan to both add new categories, e.g. on parametric timed automata as mentioned above or on entirely new problems that surface in the coming years, and also perform more extensive performance evaluations in those categories where tools will have matured sufficiently and a good benchmark set will have become available. As such, we expect a mix of "friendly" categories that stimulate tool development and standardisation as well as more "competitive" evaluations where performance really counts. Practically, we may need to split off the reports of the larger categories—those where many tools are evaluated on comprehensive benchmark sets to obtain representative performance comparisons—from the main competition report into publications of their own. In parallel to the transformation of QComp that started with this edition, the comparison of established tools on basic problems as in QComp 2019 and 2020 is likely to transition

Table 11. Data availability for QComp 2023

Section	Category	DOI	Ref.
4	Long-Run Average Rewards	10.5281/zenodo.8219191	[100]
6	Multi-Objective Analysis	10.5281/zenodo.8063883	[195]
7	Parametric Markov Models	10.5281/zenodo.10646479	[137]
8	Partially-Observable MDPs	10.5281/zenodo.8215337	[31]
9	Rare Events	10.6084/m9.figshare.23818395	[47]
11	State Space Exploration	10.5281/zenodo.10626177	[144]
12	Stochastic Games	10.5281/zenodo.7831387	[180]

into a continuous evaluation—rather than periodic competitions—hosted on the qcomp.org website. We look forward to a continuing journey into the undiscovered country beyond today's frontiers of quantitative verification in the next editions of the QComp friendly competition!

Data availability. In each category that performed a performance comparison, we provide an artifact that archives the models, tools, scripts, and other data that is necessary to reproduce the respective experiments. The benchmark set of parametric Markov models introduced in Sect. 7 is also publicly archived. We link to the DOIs of the respective datasets at the end of each of sections 4, 6, 7, 8, 9, 11, and 12, and list all of them in Table 11.

References

1. Abate, A., Andriushchenko, R., Ceska, M., Kwiatkowska, M.: Adaptive formal approximations of Markov chains. Perform. Evaluation **148**, 102207 (2021). https://doi.org/10.1016/j.peva.2021.102207
2. Agarwal, C., Guha, S., Kretínský, J., Muruganandham, P.: PAC statistical model checking of mean payoff in discrete- and continuous-time MDP. In: Shoham, S., Vizel, Y. (eds.) 34th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 13372, pp. 3–25. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_1
3. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1–6:39 (2018). https://doi.org/10.1145/3158668
4. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8
5. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002). https://doi.org/10.1145/585265.585270
6. André, É.: IMITATOR 3: Synthesis of timing parameters beyond decidability. In: Silva, A., Leino, K.R.M. (eds.) 33rd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12759, pp. 552–565. Springer (2021). https://doi.org/10.1007/978-3-030-81685-8_26

7. André, É., Marinho, D., van de Pol, J.: A benchmarks library for extended parametric timed automata. In: Loulergue, F., Wotawa, F. (eds.) 15th International Conference on Tests and Proofs (TAP). Lecture Notes in Computer Science, vol. 12740, pp. 39–50. Springer (2021). https://doi.org/10.1007/978-3-030-79379-1_3
8. Andriushchenko, R., Ceska, M., Junges, S., Katoen, J.P.: Inductive synthesis of finite-state controllers for POMDPs. In: Cussens, J., Zhang, K. (eds.) 38th Conference on Uncertainty in Artificial Intelligence (UAI). Proceedings of Machine Learning Research, vol. 180, pp. 85–95. PMLR (2022)
9. Andriushchenko, R., Ceska, M., Junges, S., Katoen, J.P., Stupinský, S.: PAYNT: A tool for inductive synthesis of probabilistic programs. In: Silva, A., Leino, K.R.M. (eds.) 33rd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12759, pp. 856–869. Springer (2021). https://doi.org/10.1007/978-3-030-81685-8_40
10. Arming, S., Bartocci, E., Chatterjee, K., Katoen, J.P., Sokolova, A.: Parameter-independent strategies for pMDPs via POMDPs. In: McIver, A., Horváth, A. (eds.) 15th International Conference on the Quantitative Evaluation of Systems (QEST). Lecture Notes in Computer Science, vol. 11024, pp. 53–70. Springer (2018). https://doi.org/10.1007/978-3-319-99154-2_4
11. Ashok, P., Brázdil, T., Kretínský, J., Slámecka, O.: Monte Carlo tree search for verifying reachability in Markov decision processes. In: Margaria, T., Steffen, B. (eds.) 8th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA). Lecture Notes in Computer Science, vol. 11245, pp. 322–335. Springer (2018). https://doi.org/10.1007/978-3-030-03421-4_21
12. Ashok, P., Chatterjee, K., Daca, P., Kretínský, J., Meggendorfer, T.: Value iteration for long-run average reward in Markov decision processes. In: Majumdar, R., Kuncak, V. (eds.) 29th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 10426, pp. 201–221. Springer (2017). https://doi.org/10.1007/978-3-319-63387-9_10
13. Azeem, M., Evangelidis, A., Kretínský, J., Slivinskiy, A., Weininger, M.: Optimistic and topological value iteration for simple stochastic games. In: Bouajjani, A., Holík, L., Wu, Z. (eds.) 20th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 13505, pp. 285–302. Springer (2022). https://doi.org/10.1007/978-3-031-19992-9_18
14. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Model-checking continous-time Markov chains. ACM Trans. Comput. Log. **1**(1), 162–170 (2000). https://doi.org/10.1145/343369.343402
15. Babiak, T., Blahoudek, F., Duret-Lutz, A., Klein, J., Kretínský, J., Müller, D., Parker, D., Strejcek, J.: The Hanoi omega-automata format. In: Kroening, D., Pasareanu, C.S. (eds.) 27th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 9206, pp. 479–486. Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_31
16. Backenköhler, M., Bortolussi, L., Großmann, G., Wolf, V.: Abstraction-guided truncations for stationary distributions of Markov population models. In: Abate, A., Marin, A. (eds.) 18th International Conference on the Quantitative Evaluation of Systems (QEST). Lecture Notes in Computer Science, vol. 12846, pp. 351–371. Springer (2021). https://doi.org/10.1007/978-3-030-85172-9_19

17. Badings, T., Simão, T.D., Suilen, M., Jansen, N.: Decision-making under uncertainty: beyond probabilities. Int. J. Softw. Tools Technol. Transf. (2023). https://doi.org/10.1007/s10009-023-00704-3
18. Badings, T.S., Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.P., Topcu, U.: Scenario-based verification of uncertain parametric MDPs. Int. J. Softw. Tools Technol. Transf. **24**(5), 803–819 (2022). https://doi.org/10.1007/s10009-022-00673-z
19. Badings, T.S., Romao, L., Abate, A., Parker, D., Poonawala, H.A., Stoelinga, M., Jansen, N.: Robust control for dynamical systems with non-Gaussian noise via formal abstractions. J. Artif. Intell. Res. **76**, 341–391 (2023). https://doi.org/10.1613/jair.1.14253
20. Baier, C., de Alfaro, L., Forejt, V., Kwiatkowska, M.: Model checking probabilistic systems. In: Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 963–999. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8_28
21. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Trans. Software Eng. **29**(6), 524–541 (2003). https://doi.org/10.1109/TSE.2003.1205180
22. Baier, C., Hensel, C., Hutschenreiter, L., Junges, S., Katoen, J.P., Klein, J.: Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination. Inf. Comput. **272**, 104504 (2020). https://doi.org/10.1016/j.ic.2019.104504
23. Bals, S., Evangelidis, A., Grover, K., Kretínský, J., Waibel, J.: MULTIGAIN 2.0: MDP controller synthesis for multiple mean-payoff, LTL and steady-state constraints. CoRR **abs/2305.16752** (2023). https://doi.org/10.48550/arXiv.2305.16752
24. Barbot, B., Haddad, S., Picaronny, C.: Coupling and importance sampling for statistical model checking. In: Flanagan, C., König, B. (eds.) 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 7214, pp. 331–346. Springer (2012). https://doi.org/10.1007/978-3-642-28756-5_23
25. Bartocci, E., Beyer, D., Black, P.E., Fedyukovich, G., Garavel, H., Hartmanns, A., Huisman, M., Kordon, F., Nagele, J., Sighireanu, M., Steffen, B., Suda, M., Sutcliffe, G., Weber, T., Yamada, A.: TOOLympics 2019: An overview of competitions in formal methods. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) 25 Years of TACAS: TOOLympics. Lecture Notes in Computer Science, vol. 11429, pp. 3–24. Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_1
26. Bartocci, E., Grosu, R., Katsaros, P., Ramakrishnan, C.R., Smolka, S.A.: Model repair for probabilistic systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 6605, pp. 326–340. Springer (2011). https://doi.org/10.1007/978-3-642-19835-9_30
27. Basset, N., Kwiatkowska, M.Z., Topcu, U., Wiltsche, C.: Strategy synthesis for stochastic games with multiple long-run objectives. In: Baier, C., Tinelli, C. (eds.) 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 9035, pp. 256–271. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_22
28. Batz, K., Junges, S., Kaminski, B.L., Katoen, J.P., Matheja, C., Schröer, P.: PrIC3: Property directed reachability for MDPs. In: Lahiri, S.K., Wang, C. (eds.) 32nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12225, pp. 512–538. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_27

29. Bellman, R.: A Markovian decision process. J. Math. Mech. **6**(5), 679–684 (1957)
30. Bohnenkamp, H.C., D'Argenio, P.R., Hermanns, H., Katoen, J.P.: MoDeST: A compositional modeling formalism for hard and softly timed systems. IEEE Trans. Software Eng. **32**(10), 812–830 (2006). https://doi.org/10.1109/TSE.2006.104
31. Bork, A.: Replication package QComp 2023 – POMDP analysis (2023). https://doi.org/10.5281/zenodo.8215337
32. Bork, A., Junges, S., Katoen, J.P., Quatmann, T.: Verification of indefinite-horizon POMDPs. In: Hung, D.V., Sokolsky, O. (eds.) 18th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 12302, pp. 288–304. Springer (2020). https://doi.org/10.1007/978-3-030-59152-6_16
33. Bork, A., Katoen, J.P., Quatmann, T.: Under-approximating expected total rewards in POMDPs. In: Fisman, D., Rosu, G. (eds.) 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 13244, pp. 22–40. Springer (2022). https://doi.org/10.1007/978-3-030-99527-0_2
34. Bortolussi, L., Silvetti, S.: Bayesian statistical parameter synthesis for linear temporal properties of stochastic models. In: Beyer, D., Huisman, M. (eds.) 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 10806, pp. 396–413. Springer (2018). https://doi.org/10.1007/978-3-319-89963-3_23
35. Brázdil, T., Brozek, V., Chatterjee, K., Forejt, V., Kucera, A.: Two views on multiple mean-payoff objectives in Markov decision processes. Log. Methods Comput. Sci. **10**(1) (2014). https://doi.org/10.2168/LMCS-10(1:13)2014
36. Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Verification of Markov decision processes using learning algorithms. In: Cassez, F., Raskin, J.F. (eds.) 12th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 8837, pp. 98–114. Springer (2014). https://doi.org/10.1007/978-3-319-11936-6_8
37. Brázdil, T., Chatterjee, K., Forejt, V., Kucera, A.: MultiGain: A controller synthesis tool for MDPs with multiple mean-payoff objectives. In: Baier, C., Tinelli, C. (eds.) 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 9035, pp. 181–187. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_12
38. Brim, L., Ceska, M., Drazan, S., Safránek, D.: Exploring parameter space of stochastic biochemical systems using quantitative model checking. In: Sharygina, N., Veith, H. (eds.) 25th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 8044, pp. 107–123. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_7
39. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: 2011 IEEE International Conference on Robotics and Automation (ICRA). pp. 723–730. IEEE (2011). https://doi.org/10.1109/ICRA.2011.5980508
40. Budde, C.E., D'Argenio, P.R., Hartmanns, A.: Automated compositional importance splitting. Sci. Comput. Program. **174**, 90–108 (2019). https://doi.org/10.1016/j.scico.2019.01.006
41. Budde, C.E., D'Argenio, P.R., Hartmanns, A., Sedwards, S.: An efficient statistical model checker for nondeterminism and rare events. Int. J. Softw. Tools Technol. Transf. **22**(6), 759–780 (2020). https://doi.org/10.1007/s10009-020-00563-2

42. Budde, C.E., D'Argenio, P.R., Monti, R.E.: Compositional construction of importance functions in fully automated importance splitting. In: Puliafito, A., Trivedi, K.S., Tuffin, B., Scarpa, M., Machida, F., Alonso, J. (eds.) 10th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS). ACM (2016). https://doi.org/10.4108/eai.25-10-2016.2266501
43. Budde, C.E., D'Argenio, P.R., Monti, R.E., Stoelinga, M.: Analysis of non-Markovian repairable fault trees through rare event simulation. Int. J. Softw. Tools Technol. Transf. **24**(5), 821–841 (2022). https://doi.org/10.1007/s10009-022-00675-x
44. Budde, C.E., Dehnert, C., Hahn, E.M., Hartmanns, A., Junges, S., Turrini, A.: JANI: Quantitative model and tool interaction. In: Legay, A., Margaria, T. (eds.) 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 10206, pp. 151–168 (2017). https://doi.org/10.1007/978-3-662-54580-5_9
45. Budde, C.E., Hartmanns, A.: Replicating RESTART with prolonged retrials: An experimental report. In: Groote, J.F., Larsen, K.G. (eds.) 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 12652, pp. 373–380. Springer (2021). https://doi.org/10.1007/978-3-030-72013-1_21
46. Budde, C.E., Hartmanns, A., Klauck, M., Kretínský, J., Parker, D., Quatmann, T., Turrini, A., Zhang, Z.: On correctness, precision, and performance in quantitative verification (QComp 2020 competition report). In: Margaria, T., Steffen, B. (eds.) 9th International Symposium on Leveraging Applications of Formal Methods (ISoLA). Lecture Notes in Computer Science, vol. 12479, pp. 216–241. Springer (2020). https://doi.org/10.1007/978-3-030-83723-5_15
47. Budde, C.E., Hartmanns, A., Ruijters, E., Volk, M., Taylor, L., Israelsen, B., Zhang, Z.: QComp 2023: formal tools for rare events (experimental reproduction package). Figshare (2023). https://doi.org/10.6084/m9.figshare.23818395
48. Budde, C.E., Ruijters, E., Stoelinga, M.: The Dynamic Fault Tree Rare Event Simulator. In: Gribaudo, M., Jansen, D.N., Remke, A. (eds.) 17th International Conference on the Quantitative Evaluation of Systems (QEST). Lecture Notes in Computer Science, vol. 12289, pp. 233–238. Springer (2020). https://doi.org/10.1007/978-3-030-59854-9_17
49. Buecherl, L., Thomas, P.J., Ahmadi, M., Jeppson, J., Gerber, A., Reiss, E., Wintead, C., Zheng, H., Zhang, Z., Myers, C.J.: A collection of biological models for the development of infinite-state stochastic model checking tools. In: 15th International Workshop on Bio-Design Automation (IWBDA). pp. 44–47 (2023)
50. Burns, B., Brock, O.: Sampling-based motion planning with sensing uncertainty. In: 2007 IEEE International Conference on Robotics and Automation (ICRA). pp. 3313–3318. IEEE (2007). https://doi.org/10.1109/ROBOT.2007.363984
51. Butkova, Y., Hartmanns, A., Hermanns, H.: A Modest approach to Markov automata. ACM Trans. Model. Comput. Simul. **31**(3), 14:1–14:34 (2021). https://doi.org/10.1145/3449355
52. Butkova, Y., Wimmer, R., Hermanns, H.: Long-run rewards for Markov automata. In: Legay, A., Margaria, T. (eds.) 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 10206, pp. 188–203 (2017). https://doi.org/10.1007/978-3-662-54580-5_11
53. Cardelli, L., Kwiatkowska, M., Laurenti, L.: A stochastic hybrid approximation for chemical kinetics based on the linear noise approximation. In: Bartocci, E., Liò,

P., Paoletti, N. (eds.) 14th International Conference on Computational Methods in Systems Biology (CMSB). Lecture Notes in Computer Science, vol. 9859, pp. 147–167. Springer (2016). https://doi.org/10.1007/978-3-319-45177-0_10

54. Ceska, M., Chau, C., Kretínský, J.: SeQuaiA: A scalable tool for semi-quantitative analysis of chemical reaction networks. In: Lahiri, S.K., Wang, C. (eds.) 32nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12224, pp. 653–666. Springer (2020). https://doi.org/10.1007/978-3-030-53288-8_32
55. Ceska, M., Kretínský, J.: Semi-quantitative abstraction and analysis of chemical reaction networks. In: Dillig, I., Tasiran, S. (eds.) 31st International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 11561, pp. 475–496. Springer (2019). https://doi.org/10.1007/978-3-030-25540-4_28
56. Chatterjee, K., Gaiser, A., Kretínský, J.: Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In: Sharygina, N., Veith, H. (eds.) 25th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 8044, pp. 559–575. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_37
57. Chatterjee, K., Henzinger, T.A.: A survey of stochastic ω-regular games. J. Comput. Syst. Sci. **78**(2), 394–413 (2012). https://doi.org/10.1016/j.jcss.2011.05.002
58. Chatterjee, K., Henzinger, T.A., Jobstmann, B., Radhakrishna, A.: Gist: A solver for probabilistic games. In: Touili, T., Cook, B., Jackson, P.B. (eds.) 22nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 6174, pp. 665–669. Springer (2010). https://doi.org/10.1007/978-3-642-14295-6_57
59. Chatterjee, K., Katoen, J.P., Mohr, S., Weininger, M., Winkler, T.: Stochastic games with lexicographic objectives. Formal Methods Syst. Des. (2023). https://doi.org/10.1007/s10703-023-00411-4
60. Chatterjee, K., Katoen, J.P., Weininger, M., Winkler, T.: Stochastic games with lexicographic reachability-safety objectives. In: Lahiri, S.K., Wang, C. (eds.) 32nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12225, pp. 398–420. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_21
61. Chatterjee, K., Kretínská, Z., Kretínský, J.: Unifying two views on multiple mean-payoff objectives in Markov decision processes. Log. Methods Comput. Sci. **13**(2) (2017). https://doi.org/10.23638/LMCS-13(2:15)2017
62. Chatterjee, K., Majumdar, R., Henzinger, T.A.: Markov decision processes with multiple objectives. In: Durand, B., Thomas, W. (eds.) 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS). Lecture Notes in Computer Science, vol. 3884, pp. 325–336. Springer (2006). https://doi.org/10.1007/11672142_26
63. Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. In: Flanagan, C., König, B. (eds.) 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 7214, pp. 315–330. Springer (2012). https://doi.org/10.1007/978-3-642-28756-5_22
64. Chen, T., Hahn, E.M., Han, T., Kwiatkowska, M.Z., Qu, H., Zhang, L.: Model repair for Markov decision processes. In: Seventh International Symposium on Theoretical Aspects of Software Engineering (TASE). pp. 85–92. IEEE Computer Society (2013). https://doi.org/10.1109/TASE.2013.20

65. Cheng, C.H., Knoll, A.C., Luttenberger, M., Buckl, C.: GAVS+: An open platform for the research of algorithmic game solving. In: Abdulla, P.A., Leino, K.R.M. (eds.) 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 6605, pp. 258–261. Springer (2011). https://doi.org/10.1007/978-3-642-19835-9_22
66. Condon, A.: On algorithms for simple stochastic games. In: Cai, J.Y. (ed.) Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, pp. 51–71. DIMACS/AMS (1990). https://doi.org/10.1090/dimacs/013/04
67. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.P., Topcu, U.: Convex optimization for parameter synthesis in MDPs. IEEE Trans. Autom. Control. **67**(12), 6333–6348 (2022). https://doi.org/10.1109/TAC.2021.3133265
68. Cubuktepe, M., Jansen, N., Junges, S., Marandi, A., Suilen, M., Topcu, U.: Robust finite-state controllers for uncertain POMDPs. In: 35th AAAI Conference on Artificial Intelligence (AAAI). pp. 11792–11800. AAAI Press (2021). https://doi.org/10.1609/aaai.v35i13.17401
69. Daigle, Bernie J., J., Roh, M.K., Gillespie, D.T., Petzold, L.R.: Automated estimation of rare event probabilities in biochemical systems. J. Chem. Phys. **134**(4) (2011). https://doi.org/10.1063/1.3522769
70. D'Argenio, P.R., Hartmanns, A., Legay, A., Sedwards, S.: Statistical approximation of optimal schedulers for probabilistic timed automata. In: Ábrahám, E., Huisman, M. (eds.) 12th International Conference on Integrated Formal Methods (iFM). Lecture Notes in Computer Science, vol. 9681, pp. 99–114. Springer (2016). https://doi.org/10.1007/978-3-319-33693-0_7
71. D'Argenio, P.R., Monti, R.E.: Input/output stochastic automata with urgency: Confluence and weak determinism. In: Fischer, B., Uustalu, T. (eds.) 15th International Colloquium on Theoretical Aspects of Computing (ICTAC). Lecture Notes in Computer Science, vol. 11187, pp. 132–152. Springer (2018). https://doi.org/10.1007/978-3-030-02508-3_8
72. David, A., Jensen, P.G., Larsen, K.G., Mikucionis, M., Taankvist, J.H.: Uppaal Stratego. In: Baier, C., Tinelli, C. (eds.) 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 9035, pp. 206–211. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_16
73. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: Liu, Z., Araki, K. (eds.) First International Colloquium on Theoretical Aspects of Computing (ICTAC). Lecture Notes in Computer Science, vol. 3407, pp. 280–294. Springer (2004). https://doi.org/10.1007/978-3-540-31862-0_21
74. Delgrange, F., Katoen, J.P., Quatmann, T., Randour, M.: Simple strategies in multi-objective MDPs. In: Biere, A., Parker, D. (eds.) 26th International Conference Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 12078, pp. 346–364. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_19
75. van Dijk, T., van de Pol, J.: Sylvan: multi-core framework for decision diagrams. Int. J. Softw. Tools Technol. Transf. **19**(6), 675–696 (2017). https://doi.org/10.1007/s10009-016-0433-2
76. Donovan, R.M., Sedgewick, A.J., Faeder, J.R., Zuckerman, D.M.: Efficient stochastic simulation of chemical kinetics networks using a weighted ensemble of trajectories. J. Chem. Phys. **139**(11) (2013). https://doi.org/10.1063/1.4821167

77. Duret-Lutz, A., Renault, E., Colange, M., Renkin, F., Aisse, A.G., Schlehuber-Caissier, P., Medioni, T., Martin, A., Dubois, J., Gillard, C., Lauko, H.: From Spot 2.0 to Spot 2.10: What's new? In: Shoham, S., Vizel, Y. (eds.) 34th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 13372, pp. 174–187. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_9
78. Egorov, M., Sunberg, Z.N., Balaban, E., Wheeler, T.A., Gupta, J.K., Kochenderfer, M.J.: POMDPs.jl: A framework for sequential decision making under uncertainty. J. Mach. Learn. Res. **18**, 26:1–26:5 (2017)
79. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: 25th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 342–351. IEEE Computer Society (2010). https://doi.org/10.1109/LICS.2010.41
80. Eisentraut, J., Kelmendi, E., Kretínský, J., Weininger, M.: Value iteration for simple stochastic games: Stopping criterion and learning algorithm. Inf. Comput. **285**(Part), 104886 (2022). https://doi.org/10.1016/j.ic.2022.104886
81. Esparza, J., Kretínský, J.: From LTL to deterministic automata: A safraless compositional approach. In: Biere, A., Bloem, R. (eds.) 26th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 8559, pp. 192–208. Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_13
82. Esparza, J., Kretínský, J., Sickert, S.: A unified translation of linear temporal logic to ω-automata. J. ACM **67**(6), 33:1–33:61 (2020). https://doi.org/10.1145/3417995
83. Etessami, K., Kwiatkowska, M.Z., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. Log. Methods Comput. Sci. **4**(4) (2008). https://doi.org/10.2168/LMCS-4(4:8)2008
84. Fedyukovich, G., Mover, S.: TACAS 23 artifact evaluation VM – Ubuntu 22.04 LTS (2022). https://doi.org/10.5281/zenodo.7113223
85. Filieri, A., Tamburrelli, G., Ghezzi, C.: Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. IEEE Trans. Software Eng. **42**(1), 75–99 (2016). https://doi.org/10.1109/TSE.2015.2421318
86. Fontanarrosa, P., Doosthosseini, H., Borujeni, A.E., Dorfan, Y., Voigt, C.A., Myers, C.: Genetic circuit dynamics: Hazard and glitch analysis. ACS Synth. Biol. **9**(9), 2324–2338 (2020). https://doi.org/10.1021/acssynbio.0c00055
87. Forejt, V., Kwiatkowska, M.Z., Norman, G., Parker, D., Qu, H.: Quantitative multi-objective verification for probabilistic systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 6605, pp. 112–127. Springer (2011). https://doi.org/10.1007/978-3-642-19835-9_11
88. Forejt, V., Kwiatkowska, M.Z., Parker, D.: Pareto curves for probabilistic model checking. In: Chakraborty, S., Mukund, M. (eds.) 10th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 7561, pp. 317–332. Springer (2012). https://doi.org/10.1007/978-3-642-33386-6_25
89. Fränzle, M., Hahn, E.M., Hermanns, H., Wolovick, N., Zhang, L.: Measurability and safety verification for stochastic hybrid systems. In: Caccamo, M., Frazzoli, E., Grosu, R. (eds.) 14th ACM International Conference on Hybrid Systems: Computation and Control (HSCC). pp. 43–52. ACM (2011). https://doi.org/10.1145/1967701.1967710

90. Frehse, G., Althoff, M. (eds.): 4th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH), EPiC Series in Computing, vol. 48. EasyChair (2017), https://easychair.org/publications/volume/ARCH17

91. Fu, C., Hahn, E.M., Li, Y., Schewe, S., Sun, M., Turrini, A., Zhang, L.: EPMC gets knowledge in multi-agent systems. In: Finkbeiner, B., Wies, T. (eds.) 23rd International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI). Lecture Notes in Computer Science, vol. 13182, pp. 93–107. Springer (2022). https://doi.org/10.1007/978-3-030-94583-1_5

92. Gainer, P., Hahn, E.M., Schewe, S.: Accelerated model checking of parametric Markov chains. In: Lahiri, S.K., Wang, C. (eds.) 16th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 11138, pp. 300–316. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_18

93. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977). https://doi.org/10.1021/j100540a008

94. Goldberg, F., Vesely, W.: Fault Tree Handbook. NUREG-0492, Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission (1981)

95. Goutsias, J.: Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. J. Chem. Phys. **122**(18) (2005). https://doi.org/10.1063/1.1889434

96. Goyal, V., Grand-Clément, J.: Robust Markov decision processes: Beyond rectangularity. Math. Oper. Res. **48**(1), 203–226 (2023). https://doi.org/10.1287/moor.2022.1259

97. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Köhl, M.A., Wolf, V.: MoGym: Using formal models for training and verifying decision-making agents. In: Shoham, S., Vizel, Y. (eds.) 34th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 13372, pp. 430–443. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_21

98. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Deep statistical model checking. In: Gotsman, A., Sokolova, A. (eds.) 40th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE). Lecture Notes in Computer Science, vol. 12136, pp. 96–114. Springer (2020). https://doi.org/10.1007/978-3-030-50086-3_6

99. Gross, D., Jansen, N., Junges, S., Pérez, G.A.: COOL-MC: A comprehensive tool for reinforcement learning and model checking. In: Dong, W., Talpin, J.P. (eds.) 8th International Symposium on Dependable Software Engineering: Theories, Tools, and Applications (SETTA). Lecture Notes in Computer Science, vol. 13649, pp. 41–49. Springer (2022). https://doi.org/10.1007/978-3-031-21213-0_3

100. Grover, K.: QComp LRA results (2023). https://doi.org/10.5281/zenodo.8219191

101. Guck, D., Timmer, M., Hatefi, H., Ruijters, E., Stoelinga, M.: Modelling and analysis of Markov reward automata. In: Cassez, F., Raskin, J.F. (eds.) 12th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 8837, pp. 168–184. Springer (2014). https://doi.org/10.1007/978-3-319-11936-6_13

102. Hahn, E.M., Hartmanns, A.: Symblicit exploration and elimination for probabilistic model checking. In: Hung, C.C., Hong, J., Bechini, A., Song, E. (eds.) 36th ACM/SIGAPP Symposium on Applied Computing (SAC). pp. 1798–1806. ACM (2021). https://doi.org/10.1145/3412841.3442052

103. Hahn, E.M., Hartmanns, A., Hensel, C., Klauck, M., Klein, J., Kretínský, J., Parker, D., Quatmann, T., Ruijters, E., Steinmetz, M.: The 2019 comparison of tools for the analysis of quantitative formal models (QComp 2019 competition report). In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) 25 Years of TACAS: TOOLympics. Lecture Notes in Computer Science, vol. 11429, pp. 69–92. Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_5
104. Hahn, E.M., Hartmanns, A., Hermanns, H.: Reachability and reward checking for stochastic timed automata. Electron. Commun. Eur. Assoc. Softw. Sci. Technol. **70** (2014). https://doi.org/10.14279/tuj.eceasst.70.968
105. Hahn, E.M., Hartmanns, A., Hermanns, H., Katoen, J.P.: A compositional modelling and analysis framework for stochastic hybrid systems. Formal Methods Syst. Des. **43**(2), 191–232 (2013). https://doi.org/10.1007/s10703-012-0167-z
106. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: INFAMY: An infinite-state Markov model checker. In: Bouajjani, A., Maler, O. (eds.) 21st International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 5643, pp. 641–647. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_49
107. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. Int. J. Softw. Tools Technol. Transf. **13**(1), 3–19 (2011). https://doi.org/10.1007/s10009-010-0146-x
108. Hahn, E.M., Li, G., Schewe, S., Turrini, A., Zhang, L.: Lazy probabilistic model checking without determinisation. In: Aceto, L., de Frutos-Escrig, D. (eds.) 26th International Conference on Concurrency Theory (CONCUR). LIPIcs, vol. 42, pp. 354–367. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2015). https://doi.org/10.4230/LIPIcs.CONCUR.2015.354
109. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: iscasMc: A web-based probabilistic model checker. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) 19th International Symposium on Formal Methods (FM). Lecture Notes in Computer Science, vol. 8442, pp. 312–317. Springer (2014). https://doi.org/10.1007/978-3-319-06410-9_22
110. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In: Biere, A., Parker, D. (eds.) 26th International Conference Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 12078, pp. 306–323. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_17
111. Hartmanns, A.: Correct probabilistic model checking with floating-point arithmetic. In: Fisman, D., Rosu, G. (eds.) 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 13244, pp. 41–59. Springer (2022). https://doi.org/10.1007/978-3-030-99527-0_3
112. Hartmanns, A., Hermanns, H.: A Modest approach to checking probabilistic timed automata. In: 6th International Conference on the Quantitative Evaluation of Systems (QEST). pp. 187–196. IEEE Computer Society (2009). https://doi.org/10.1109/QEST.2009.41
113. Hartmanns, A., Hermanns, H.: The Modest Toolset: An integrated environment for quantitative modelling and verification. In: Ábrahám, E., Havelund, K. (eds.) 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 8413, pp. 593–598. Springer (2014). https://doi.org/10.1007/978-3-642-54862-8_51

114. Hartmanns, A., Hermanns, H.: Explicit model checking of very large MDP using partitioning and secondary storage. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) 13th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 9364, pp. 131–147. Springer (2015). https://doi.org/10.1007/978-3-319-24953-7_10

115. Hartmanns, A., Junges, S., Katoen, J.P., Quatmann, T.: Multi-cost bounded tradeoff analysis in MDP. J. Autom. Reason. **64**(7), 1483–1522 (2020). https://doi.org/10.1007/s10817-020-09574-9

116. Hartmanns, A., Junges, S., Quatmann, T., Weininger, M.: A practitioner's guide to MDP model checking algorithms. In: Sankaranarayanan, S., Sharygina, N. (eds.) 29th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 13993, pp. 469–488. Springer (2023). https://doi.org/10.1007/978-3-031-30823-9_24

117. Hartmanns, A., Kaminski, B.L.: Optimistic value iteration. In: Lahiri, S.K., Wang, C. (eds.) 32nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12225, pp. 488–511. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_26

118. Hartmanns, A., Katoen, J.P., Kohlen, B., Spel, J.: Tweaking the odds in probabilistic timed automata. In: Abate, A., Marin, A. (eds.) 18th International Conference on the Quantitative Evaluation of Systems (QEST). Lecture Notes in Computer Science, vol. 12846, pp. 39–58. Springer (2021). https://doi.org/10.1007/978-3-030-85172-9_3

119. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 11427, pp. 344–350. Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_20

120. Hartmanns, A., Sedwards, S., D'Argenio, P.R.: Efficient simulation-based verification of probabilistic timed automata. In: 2017 Winter Simulation Conference (WSC). pp. 1419–1430. IEEE (2017). https://doi.org/10.1109/WSC.2017.8247885

121. Hasenauer, J., Wolf, V., Kazeroonian, A., Theis, F.J.: Method of conditional moments (MCM) for the chemical master equation. J. Math. Biol. **69**(3), 687–735 (2013). https://doi.org/10.1007/s00285-013-0711-5

122. Heck, L., Spel, J., Junges, S., Moerman, J., Katoen, J.P.: Gradient-descent for randomized controllers under partial observability. In: Finkbeiner, B., Wies, T. (eds.) 23rd International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI). Lecture Notes in Computer Science, vol. 13182, pp. 127–150. Springer (2022). https://doi.org/10.1007/978-3-030-94583-1_7

123. Heidelberger, P.: Fast simulation of rare events in queueing and reliability models. In: Donatiello, L., Nelson, R.D. (eds.) Performance Evaluation of Computer and Communication Systems – Joint Tutorial Papers of Performance '93 and Sigmetrics '93. Lecture Notes in Computer Science, vol. 729, pp. 165–202. Springer (1993). https://doi.org/10.1007/BFb0013853

124. Helfrich, M., Ceska, M., Kretínský, J., Marticek, S.: Abstraction-based segmental simulation of chemical reaction networks. In: Petre, I., Paun, A. (eds.) 20th International Conference on Computational Methods in Systems Biology (CMSB). Lecture Notes in Computer Science, vol. 13447, pp. 41–60. Springer (2022). https://doi.org/10.1007/978-3-031-15034-0_3

125. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker Storm. Int. J. Softw. Tools Technol. Transf. **24**(4), 589–610 (2022). https://doi.org/10.1007/s10009-021-00633-z
126. Henzinger, T.A., Mikeev, L., Mateescu, M., Wolf, V.: Hybrid numerical solution of the chemical master equation. In: Quaglia, P. (ed.) 8th International Conference on Computational Methods in Systems Biology (CMSB). pp. 55–65. ACM (2010). https://doi.org/10.1145/1839764.1839772
127. Hermanns, H., Meyer-Kayser, J., Siegle, M.: Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In: Plateau, B., Stewart, W., Silva, M. (eds.) 3rd International Workshop on Numerical Solution of Markov Chains (NSMC). pp. 188–207. Prensas Universitarias de Zaragoza (1999)
128. Holtzen, S., Junges, S., Vazquez-Chanlatte, M., Millstein, T.D., Seshia, S.A., den Broeck, G.V.: Model checking finite-horizon Markov chains with probabilistic inference. In: Silva, A., Leino, K.R.M. (eds.) 33rd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12760, pp. 577–601. Springer (2021). https://doi.org/10.1007/978-3-030-81688-9_27
129. Israelsen, B., Taylor, L., Zhang, Z.: Efficient trace generation for rare-event analysis in chemical reaction networks. In: Caltais, G., Schilling, C. (eds.) 29th International Symposium on Model Checking Software (SPIN). Lecture Notes in Computer Science, vol. 13872, pp. 83–102. Springer (2023). https://doi.org/10.1007/978-3-031-32157-3_5
130. Itoh, H., Nakamura, K.: Partially observable Markov decision processes with imprecise parameters. Artif. Intell. **171**(8-9), 453–490 (2007). https://doi.org/10.1016/j.artint.2007.03.004
131. Jackson, J.R.: Networks of waiting lines. Operations Research **5**, 518–521 (1957)
132. Jansen, N., Junges, S., Katoen, J.P.: Parameter synthesis in Markov models: A gentle survey. In: Raskin, J.F., Chatterjee, K., Doyen, L., Majumdar, R. (eds.) Principles of Systems Design – Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 13660, pp. 407–437. Springer (2022). https://doi.org/10.1007/978-3-031-22337-2_20
133. Jégourel, C., Legay, A., Sedwards, S.: Command-based importance sampling for statistical model checking. Theor. Comput. Sci. **649**, 1–24 (2016). https://doi.org/10.1016/j.tcs.2016.08.009
134. Jeppson, J., Volk, M., Israelsen, B., Roberts, R., Williams, A., Buecherl, L., Myers, C.J., Zheng, H., Winstead, C., Zhang, Z.: STAMINA in C++: Modernizing an infinite-state probabilistic model checker. In: Jansen, N., Tribastone, M. (eds.) 20th International Conference on the Quantitative Evaluation of Systems (QEST). Lecture Notes in Computer Science, vol. 14287, pp. 101–109. Springer (2023). https://doi.org/10.1007/978-3-031-43835-6_7
135. John, T., Jantsch, S., Baier, C., Klüppelholz, S.: From Emerson-Lei automata to deterministic, limit-deterministic or good-for-MDP automata. Innov. Syst. Softw. Eng. **18**(3), 385–403 (2022). https://doi.org/10.1007/s11334-022-00445-7
136. Junges, S.: Parameter synthesis in Markov models. Ph.D. thesis, RWTH Aachen University (2020), https://publications.rwth-aachen.de/record/783179
137. Junges, S.: sjunges/parametric-Markov-models: 0.2 (2023). https://doi.org/10.5281/zenodo.10646479
138. Junges, S., Ábrahám, E., Hensel, C., Jansen, N., Katoen, J.P., Quatmann, T., Volk, M.: Parameter synthesis for Markov models. CoRR **abs/1903.07993** (2019). https://doi.org/10.48550/arXiv.1903.07993

139. Junges, S., Jansen, N., Seshia, S.A.: Enforcing almost-sure reachability in POMDPs. In: Silva, A., Leino, K.R.M. (eds.) 33rd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12760, pp. 602–625. Springer (2021). https://doi.org/10.1007/978-3-030-81688-9_28
140. Junges, S., Spaan, M.T.J.: Abstraction-refinement for hierarchical probabilistic models. In: Shoham, S., Vizel, Y. (eds.) 34th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 13371, pp. 102–123. Springer (2022). https://doi.org/10.1007/978-3-031-13185-1_6
141. Kahn, H., Harris, T.E.: Estimation of particle transmission by random sampling. National Bureau of Standards Applied Mathematics Series **12**, 27–30 (1951)
142. Klauck, M., Steinmetz, M., Hoffmann, J., Hermanns, H.: Bridging the gap between probabilistic model checking and probabilistic planning: Survey, compilations, and empirical comparison. J. Artif. Intell. Res. **68**, 247–310 (2020). https://doi.org/10.1613/jair.1.11595
143. Kochenderfer, M.J.: Decision Making Under Uncertainty: Theory and Application. MIT Press (2015)
144. Köhl, M.A.: QComp 2023: State space exploration artifact (2024). https://doi.org/10.5281/zenodo.10626177
145. Köhl, M.A., Klauck, M., Hermanns, H.: Momba: JANI meets Python. In: Groote, J.F., Larsen, K.G. (eds.) 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 12652, pp. 389–398. Springer (2021). https://doi.org/10.1007/978-3-030-72013-1_23
146. Kretínský, J.: LTL-constrained steady-state policy synthesis. In: Zhou, Z.H. (ed.) 30th International Joint Conference on Artificial Intelligence (IJCAI). pp. 4104–4111. ijcai.org (2021). https://doi.org/10.24963/ijcai.2021/565
147. Kretínský, J., Esparza, J.: Deterministic automata for the (f, g)-fragment of LTL. In: Madhusudan, P., Seshia, S.A. (eds.) 24th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 7358, pp. 7–22. Springer (2012). https://doi.org/10.1007/978-3-642-31424-7_7
148. Kretínský, J., Meggendorfer, T.: Of cores: A partial-exploration framework for Markov decision processes. Log. Methods Comput. Sci. **16**(4) (2020). https://doi.org/10.23638/LMCS-16(4:3)2020
149. Kretínský, J., Meggendorfer, T., Sickert, S.: LTL Store: Repository of LTL formulae from literature and case studies. CoRR **abs/1807.03296** (2018). https://doi.org/10.48550/arXiv.1807.03296
150. Kretínský, J., Meggendorfer, T., Sickert, S.: Owl: A library for ω-words, automata, and LTL. In: Lahiri, S.K., Wang, C. (eds.) 16th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 11138, pp. 543–550. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_34
151. Kretínský, J., Meggendorfer, T., Sickert, S., Ziegler, C.: Rabinizer 4: From LTL to your favourite deterministic automaton. In: Chockler, H., Weissenbacher, G. (eds.) 30th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 10981, pp. 567–577. Springer (2018). https://doi.org/10.1007/978-3-319-96145-3_30
152. Kretínský, J., Meggendorfer, T., Weininger, M.: Stopping criteria for value iteration on stochastic games with quantitative objectives. In: 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–14 (2023). https://doi.org/10.1109/LICS56636.2023.10175771

153. Kretínský, J., Michel, F., Michel, L., Pérez, G.A.: Finite-memory near-optimal learning for Markov decision processes with long-run average reward. In: Adams, R.P., Gogate, V. (eds.) 36th Conference on Uncertainty in Artificial Intelligence (UAI). Proceedings of Machine Learning Research, vol. 124, pp. 1149–1158. AUAI Press (2020)
154. Kretínský, J., Ramneantu, E., Slivinskiy, A., Weininger, M.: Comparison of algorithms for simple stochastic games. Inf. Comput. **289**(Part), 104885 (2022). https://doi.org/10.1016/j.ic.2022.104885
155. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Brock, O., Trinkle, J., Ramos, F. (eds.) Robotics: Science and Systems IV. The MIT Press (2008). https://doi.org/10.15607/RSS.2008.IV.009
156. Kurose, J.F., Ross, K.W.: Computer networking – a top-down approach featuring the Internet. Addison-Wesley-Longman (2001)
157. Kuwahara, H., Mura, I.: An efficient and exact stochastic simulation method to analyze rare events in biochemical systems. J. Chem. Phys. **129**(16) (2008). https://doi.org/10.1063/1.2987701
158. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In: Lahiri, S.K., Wang, C. (eds.) 32nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12225, pp. 475–487. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_25
159. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Automatic verification of concurrent stochastic systems. Formal Methods Syst. Des. **58**(1-2), 188–250 (2021). https://doi.org/10.1007/s10703-020-00356-y
160. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Correlated equilibria and fairness in concurrent stochastic games. In: Fisman, D., Rosu, G. (eds.) 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 13244, pp. 60–78. Springer (2022). https://doi.org/10.1007/978-3-030-99527-0_4
161. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Symbolic verification and strategy synthesis for turn-based stochastic games. In: Raskin, J.F., Chatterjee, K., Doyen, L., Majumdar, R. (eds.) Principles of Systems Design – Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 13660, pp. 388–406. Springer (2022). https://doi.org/10.1007/978-3-031-22337-2_19
162. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: Ouaknine, J., Vaandrager, F.W. (eds.) 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes in Computer Science, vol. 5813, pp. 212–227. Springer (2009). https://doi.org/10.1007/978-3-642-04368-0_17
163. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) 23rd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_47
164. Kwiatkowska, M.Z., Norman, G., Parker, D.: The PRISM benchmark suite. In: 9th International Conference on the Quantitative Evaluation of Systems (QEST). pp. 203–204. IEEE Computer Society (2012). https://doi.org/10.1109/QEST.2012.14

165. Kwiatkowska, M.Z., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Formal Methods Syst. Des. **29**(1), 33–78 (2006). https://doi.org/10.1007/s10703-006-0005-2

166. Kwiatkowska, M.Z., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. **282**(1), 101–150 (2002). https://doi.org/10.1016/S0304-3975(01)00046-9

167. Kwiatkowska, M.Z., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Inf. Comput. **205**(7), 1027–1077 (2007). https://doi.org/10.1016/j.ic.2007.01.004

168. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Parametric probabilistic transition systems for system design and analysis. Formal Aspects Comput. **19**(1), 93–109 (2007). https://doi.org/10.1007/s00165-006-0015-2

169. Legay, A., Lukina, A., Traonouez, L.M., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Steffen, B., Woeginger, G.J. (eds.) Computing and Software Science – State of the Art and Perspectives, Lecture Notes in Computer Science, vol. 10000, pp. 478–504. Springer (2019). https://doi.org/10.1007/978-3-319-91908-9_23

170. Li, M., Turrini, A., Hahn, E.M., She, Z., Zhang, L.: Probabilistic preference planning problem for Markov decision processes. IEEE Trans. Software Eng. **48**(5), 1545–1559 (2022). https://doi.org/10.1109/TSE.2020.3024215

171. Lovejoy, W.S.: Computationally feasible bounds for partially observed Markov decision processes. Oper. Res. **39**(1), 162–175 (1991). https://doi.org/10.1287/opre.39.1.162

172. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and related stochastic optimization problems. Artif. Intell. **147**(1-2), 5–34 (2003). https://doi.org/10.1016/S0004-3702(02)00378-8

173. Madsen, C., Zhang, Z., Roehner, N., Winstead, C., Myers, C.J.: Stochastic model checking of genetic circuits. ACM J. Emerg. Technol. Comput. Syst. **11**(3), 23:1–23:21 (2014). https://doi.org/10.1145/2644817

174. Major, J., Blahoudek, F., Strejcek, J., Sasaráková, M., Zboncáková, T.: ltl3tela: LTL to small deterministic or nondeterministic Emerson-Lei automata. In: Chen, Y.F., Cheng, C.H., Esparza, J. (eds.) 17th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 11781, pp. 357–365. Springer (2019). https://doi.org/10.1007/978-3-030-31784-3_21

175. Mannor, S., Simester, D., Sun, P., Tsitsiklis, J.N.: Bias and variance approximation in value function estimates. Manag. Sci. **53**(2), 308–322 (2007). https://doi.org/10.1287/mnsc.1060.0614

176. Mausam, Kolobov, A.: Planning with Markov Decision Processes: An AI Perspective. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012). https://doi.org/10.2200/S00426ED1V01Y201206AIM017

177. McMillan, K.L., Zuck, L.D.: Compositional testing of Internet protocols. In: 2019 IEEE Secure Development Conference (SecDev). pp. 161–174. IEEE (2019). https://doi.org/10.1109/SecDev.2019.00031

178. Mediouni, B.L., Nouri, A., Bozga, M., Dellabani, M., Legay, A., Bensalem, S.: *S* BIP 2.0: Statistical model checking stochastic real-time systems. In: Lahiri, S.K., Wang, C. (eds.) 16th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 11138, pp. 536–542. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_33

179. Meggendorfer, T.: PET – a partial exploration tool for probabilistic verification. In: Bouajjani, A., Holík, L., Wu, Z. (eds.) 20th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 13505, pp. 320–326. Springer (2022). https://doi.org/10.1007/978-3-031-19992-9_20
180. Meggendorfer, T.: QComp 2023: Stochastic games – evaluation (2023). https://doi.org/10.5281/zenodo.7831387
181. Müller, D., Sickert, S.: LTL to deterministic Emerson-Lei automata. In: Bouyer, P., Orlandini, A., Pietro, P.S. (eds.) 8th International Symposium on Games, Automata, Logics and Formal Verification (GandALF). EPTCS, vol. 256, pp. 180–194 (2017). https://doi.org/10.4204/EPTCS.256.13
182. Munsky, B., Khammash, M.: The finite state projection algorithm for the solution of the chemical master equation. J. Chem. Phys. **124**(4) (2006). https://doi.org/10.1063/1.2145882
183. Neupane, T., Myers, C.J., Madsen, C., Zheng, H., Zhang, Z.: STAMINA: Stochastic approximate model-checker for infinite-state analysis. In: Dillig, I., Tasiran, S. (eds.) 31st International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 11561, pp. 540–549. Springer (2019). https://doi.org/10.1007/978-3-030-25540-4_31
184. Neupane, T., Zhang, Z., Madsen, C., Zheng, H., Myers, C.J.: Approximation techniques for stochastic analysis of biological systems. In: Liò, P., Zuliani, P. (eds.) Automated Reasoning for Systems Biology and Medicine, Computational Biology, vol. 30, pp. 327–348. Springer (2019). https://doi.org/10.1007/978-3-030-17297-8_12
185. Nicola, V.F., Shahabuddin, P., Nakayama, M.K.: Techniques for fast simulation of models of highly dependable systems. IEEE Trans. Reliab. **50**(3), 246–264 (2001). https://doi.org/10.1109/24.974122
186. Niehage, M., Hartmanns, A., Remke, A.: Learning optimal decisions for stochastic hybrid systems. In: Arun-Kumar, S., Méry, D., Saha, I., Zhang, L. (eds.) 19th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE). pp. 44–55. ACM (2021). https://doi.org/10.1145/3487212.3487339
187. Nilim, A., Ghaoui, L.E.: Robust control of Markov decision processes with uncertain transition matrices. Oper. Res. **53**(5), 780–798 (2005). https://doi.org/10.1287/opre.1050.0216
188. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic systems. Real Time Syst. **53**(3), 354–402 (2017). https://doi.org/10.1007/s11241-017-9269-4
189. Pai, G.J., Dugan, J.B.: Automatic synthesis of dynamic fault trees from UML system models. In: 13th International Symposium on Software Reliability Engineering (ISSRE). pp. 243–256. IEEE Computer Society (2002). https://doi.org/10.1109/ISSRE.2002.1173261
190. Phalakarn, K., Takisaka, T., Haas, T., Hasuo, I.: Widest paths and global propagation in bounded value iteration for stochastic games. In: Lahiri, S.K., Wang, C. (eds.) 32nd International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 12225, pp. 349–371. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_19
191. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (FOCS). pp. 46–57. IEEE Computer Society (1977). https://doi.org/10.1109/SFCS.1977.32

192. Pranger, S., Könighofer, B., Posch, L., Bloem, R.: TEMPEST – synthesis tool for reactive systems and shields in probabilistic environments. In: Hou, Z., Ganesh, V. (eds.) 19th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 12971, pp. 222–228. Springer (2021). https://doi.org/10.1007/978-3-030-88885-5_15
193. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In: Sharygina, N., Veith, H. (eds.) 25th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 8044, pp. 527–542. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_35
194. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994). https://doi.org/10.1002/9780470316887
195. Quatmann, T.: Replication package: QComp 2023 – multi-objective analysis (2023). https://doi.org/10.5281/zenodo.8063883
196. Quatmann, T., Junges, S., Katoen, J.P.: Markov automata with multiple objectives. Formal Methods Syst. Des. **60**(1), 33–86 (2022). https://doi.org/10.1007/s10703-021-00364-6
197. Quatmann, T., Katoen, J.P.: Multi-objective optimization of long-run average and total rewards. In: Groote, J.F., Larsen, K.G. (eds.) 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 12651, pp. 230–249. Springer (2021). https://doi.org/10.1007/978-3-030-72016-2_13
198. Reijsbergen, D., de Boer, P.T., Scheinhardt, W.R.W., Juneja, S.: Path-ZVA: General, efficient, and automated importance sampling for highly reliable Markovian systems. ACM Trans. Model. Comput. Simul. **28**(3), 22:1–22:25 (2018). https://doi.org/10.1145/3161569
199. Roberts, R., Neupane, T., Buecherl, L., Myers, C.J., Zhang, Z.: STAMINA 2.0: Improving scalability of infinite-state stochastic model checking. In: Finkbeiner, B., Wies, T. (eds.) 23rd International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI). Lecture Notes in Computer Science, vol. 13182, pp. 319–331. Springer (2022). https://doi.org/10.1007/978-3-030-94583-1_16
200. Ruijters, E., Reijsbergen, D., de Boer, P.T., Stoelinga, M.: Rare event simulation for dynamic fault trees. Reliab. Eng. Syst. Saf. **186**, 220–231 (2019). https://doi.org/10.1016/j.ress.2019.02.004
201. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (4th Edition). Pearson (2020)
202. Salmani, B., Katoen, J.P.: Fine-tuning the odds in Bayesian networks. In: Vejnarová, J., Wilson, N. (eds.) 16th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU). Lecture Notes in Computer Science, vol. 12897, pp. 268–283. Springer (2021). https://doi.org/10.1007/978-3-030-86772-0_20
203. Schwartz, A.: A reinforcement learning method for maximizing undiscounted rewards. In: Utgoff, P.E. (ed.) 10th International Conference on Machine Learning (ICML). pp. 298–305. Morgan Kaufmann (1993). https://doi.org/10.1016/b978-1-55860-307-3.50045-9
204. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based POMDP solvers. Auton. Agents Multi Agent Syst. **27**(1), 1–51 (2013). https://doi.org/10.1007/s10458-012-9200-2

205. Sickert, S., Esparza, J., Jaax, S., Kretínský, J.: Limit-deterministic Büchi automata for linear temporal logic. In: Chaudhuri, S., Farzan, A. (eds.) 28th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 9780, pp. 312–332. Springer (2016). https://doi.org/10.1007/978-3-319-41540-6_17
206. Sickert, S., Kretínský, J.: MoChiBA: Probabilistic LTL model checking using limit-deterministic Büchi automata. In: Artho, C., Legay, A., Peled, D. (eds.) 14th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 9938, pp. 130–137 (2016). https://doi.org/10.1007/978-3-319-46520-3_9
207. Spel, J., Junges, S., Katoen, J.P.: Are parametric Markov chains monotonic? In: Chen, Y.F., Cheng, C.H., Esparza, J. (eds.) 17th International Symposium on Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, vol. 11781, pp. 479–496. Springer (2019). https://doi.org/10.1007/978-3-030-31784-3_28
208. Spel, J., Junges, S., Katoen, J.P.: Finding provably optimal Markov chains. In: Groote, J.F., Larsen, K.G. (eds.) 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, vol. 12651, pp. 173–190. Springer (2021). https://doi.org/10.1007/978-3-030-72016-2_10
209. Suilen, M., Jansen, N., Cubuktepe, M., Topcu, U.: Robust policy synthesis for uncertain POMDPs via convex optimization. In: Bessiere, C. (ed.) 29th International Joint Conference on Artificial Intelligence (IJCAI). pp. 4113–4120. ijcai.org (2020). https://doi.org/10.24963/ijcai.2020/569
210. Suilen, M., Simão, T.D., Parker, D., Jansen, N.: Robust anytime learning of Markov decision processes. In: NeurIPS (2022)
211. Taylor, L., Israelsen, B., Zhang, Z.: Cycle and commute: Rare-event probability verification for chemical reaction networks. In: Nadel, A., Rozier, K.Y. (eds.) 23rd Conference on Formal Methods in Computer-Aided Design (FMCAD). pp. 284–293. TU Wien Academic Press (2023). https://doi.org/10.34727/2023/ISBN.978-3-85448-060-0_37
212. Van Kampen, N.G.: Stochastic processes in physics and chemistry, vol. 1. Elsevier (1992)
213. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification (preliminary report). In: 1st Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 332–344. IEEE Computer Society (1986)
214. Velasquez, A., Alkhouri, I., Beckus, A., Trivedi, A., Atia, G.K.: Controller synthesis for omega-regular and steady-state specifications. In: Faliszewski, P., Mascardi, V., Pelachaud, C., Taylor, M.E. (eds.) 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 1310–1318. International Foundation for Autonomous Agents and Multiagent Systems (2022). https://doi.org/10.5555/3535850.3535996
215. Villén-Altamirano, J.: RESTART vs splitting: A comparative study. Perform. Evaluation **121-122**, 38–47 (2018). https://doi.org/10.1016/j.peva.2018.02.002
216. Volk, M., Junges, S., Katoen, J.P.: Advancing dynamic fault tree analysis – get succinct state spaces fast and synthesise failure rates. In: Skavhaug, A., Guiochet, J., Bitsch, F. (eds.) 35th International Conference on Computer Safety, Reliability, and Security (SAFECOMP). Lecture Notes in Computer Science, vol. 9922, pp. 253–265. Springer (2016). https://doi.org/10.1007/978-3-319-45477-1_20

217. Volk, M., Junges, S., Katoen, J.P.: Fast dynamic fault tree analysis by model checking techniques. IEEE Trans. Ind. Informatics **14**(1), 370–379 (2018). https://doi.org/10.1109/TII.2017.2710316
218. Wiesemann, W., Kuhn, D., Sim, M.: Distributionally robust convex optimization. Oper. Res. **62**(6), 1358–1376 (2014). https://doi.org/10.1287/opre.2014.1314
219. Winkler, T., Junges, S., Pérez, G.A., Katoen, J.P.: On the complexity of reachability in parametric Markov decision processes. In: Fokkink, W.J., van Glabbeek, R. (eds.) 30th International Conference on Concurrency Theory (CONCUR). LIPIcs, vol. 140, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.CONCUR.2019.14
220. Wolff, E.M., Topcu, U., Murray, R.M.: Robust control of uncertain Markov decision processes with temporal logic specifications. In: 51th IEEE Conference on Decision and Control (CDC). pp. 3372–3379. IEEE (2012). https://doi.org/10.1109/CDC.2012.6426174
221. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Brinksma, E., Larsen, K.G. (eds.) 14th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 2404, pp. 223–235. Springer (2002). https://doi.org/10.1007/3-540-45657-0_17
222. Yu, H., Bertsekas, D.P.: Discretized approximations for POMDP with average cost. In: Chickering, D.M., Halpern, J.Y. (eds.) 20th Conference on Uncertainty in Artificial Intelligence (UAI). p. 519. AUAI Press (2004)
223. Zhang, J., Watson, L.T., Cao, Y.: Adaptive aggregation method for the chemical master equation. Int. J. Comput. Biol. Drug Des. **2**(2), 134–148 (2009). https://doi.org/10.1504/IJCBDD.2009.028825