From Verification to Synthesis under Cost-Utility Constraints

CHRISTEL BAIER and CLEMENS DUBSLAFF, Technische Universität Dresden, Germany

Various algorithms and tools for the formal verification of systems with respect to their quantitative behavior have been developed in the past decades. Many of these techniques inherently support the automated synthesis of strategies that guarantee the satisfaction of performance or reliability constraints by resolving controllable nondeterministic choices in an adequate way. More recently, such techniques have been further developed towards the analysis or strategy synthesis under multiple cost and utility constraints.

This article deals with Markov decision processes (MDPs) for modeling systems and their environment and provides an overview of recent directions for different types of synthesis problems in MDPs that aim to achieve an acceptable cost-utility tradeoff.

1 INTRODUCTION

Markov decision processes (MDPs, see, e.g., [65, 105]), are a prominent stochastic model that has been introduced in the 1950s and widely used for various types of optimization problems with applications, e.g., in operations research, reinforcement learning, and robotics. Since the 1990s, MDPs have been used as an operational model for distributed probabilistic systems and various verification algorithms for MDPs and temporal logics have been developed. In this setting, the nondeterministic choices are typically viewed as choices made by an adversary and the classical verification task is to show that a threshold condition for the probability of a temporal path property or an expectation constraint holds, no matter how the nondeterministic choices are resolved. For many types of temporal properties or expectations, worst-case strategies for resolving the nondeterministic choices exist in the sense that they maximize or minimize the probabilities resp. expectations under consideration. Indeed, many verification algorithms inherently compute such extremal strategies, which turns the classical verification problem for MDPs into a strategy-synthesis problem. In the past decade, there has been a drift from classical "single-objective" verification or synthesis problems to objectives that combine cost and utility constraints, defined, e.g., through weight functions, temporal logics, or combinations thereof. As a matter of fact, even the traditional shortest-path problem for MDPs [21] is inherently multi-objective by combining a qualitative utility condition "reaching the target almost surely" with the cost objective "minimize the expected total cost". In the past, several variants of shortest-path problems and other analysis and synthesis problems under tradeoff constraints to balance multiple cost and utility objectives have been considered. Following [40], quantitative measures for how well does a system satisfy a specification can be classified as follows:

(1) measures along behaviors, such as the cost or reward until reaching a target or the long-run average cost or reward (e.g., mean payoff, long-run cost-utility ratios),
(2) measures across behaviors, such as worst- or average-case behaviors, but also quantiles or conditional probabilities and expectations.

The development of algorithms, complexity-theoretic considerations as well as the investigation of resource requirement for worst- or best-case strategies according to specifications combining measures of type (1) and/or (2) are very active research fields.

The authors are supported by the DFG through the collaborative research centre HAEC (SFB 912) and projects BA 1679/11-1 and 1679/12-1.

Authors' address: Christel Baier, christel.baier@tu-dresden.de; Clemens Dubslaff, clemens.dubslaff@tu-dresden.de, Technische Universität Dresden, Germany.

In this article, we provide a summary of classical results and a selection of more advanced recent developments in these directions. We start in Section 2 with some basic concepts and recent achievements on how to solve strategy-synthesis problems. Other types of synthesis problems for Markovian models will be addressed in Sections 3 and 4. While Section 3 gives an overview on variant-selection problems using feature-oriented formalisms, Section 4 provides insights in the potential of parameter-synthesis approaches for Markovian models with parametric transition probabilities.

2 WORST-CASE ANALYSIS AND STRATEGY SYNTHESIS

To explain the main ideas of synthesis approaches, we briefly introduce some basic notations for *Markov decision processes (MDPs)* where we restrict ourselves to MDPs over a finite state space and action set. Details can be found in textbooks such as [65, 105] or in [7, 13] for details on the use of MDPs as an operational system model and algorithms for the formal analysis against temporal-logic specifications. We use standard linear temporal logic (LTL) like notations to denote path properties, e.g., $\Diamond G$ for the reachability condition "eventually visit a G-state" or $\Box\Diamond G$ for the Büchi condition "infinitely often G" or $\Diamond\Box G$ for the co-Büchi condition "almost always G".

Given a finite set S, a distribution on S is a function $\mu\colon S \to [0,1]$ with $\sum_{s\in S} \mu(s) = 1$. We write $\mathrm{Distr}(S)$ for the set of distributions μ on S where $\mu(s) \in \mathbb{Q}$ for all $s \in S$. A (plain) MDP is a tuple $\mathcal{M} = (S, Act, P)$ where S is a finite set of states, Act a finite set of actions, and $P\colon S \times Act \to \mathrm{Distr}(S)$ a partial probability function.

For $s \in S$ we denote by $Act(s) = \{\alpha \in Act : P(s,\alpha) \neq \bot\}$ the set of actions that are enabled in state s, where $\bot$ indicates "undefined". State s is called a trap if $Act(s) = \varnothing$. The special case of a (finite-state) *Markov chain* is obtained when Act is a singleton, in which case the action name is irrelevant and P can be treated as a partial function $P\colon S \times S \to [0,1] \cap \mathbb{Q}$.

An *end component* of $\mathcal{M}$ is a strongly connected sub-MDP. They can be specified by sets of state-action pairs. While the number of end components can be exponential, the number of maximal end components (i.e., end components that are not contained in larger end components) is bounded by $|S|$. Maximal end components are computable in polynomial time [39, 54].

A *finite path in* $\mathcal{M}$ is a sequence $\pi = s_0\,\alpha_0\,s_1\alpha_1 \ldots \alpha_{n-1}s_n$ where $s_0, s_1, \ldots, s_n \in S$ are states and the α_i's are actions with $\alpha_i \in Act(s_i)$ and $P(s_i, \alpha_i, s_{i+1}) > 0$ for all $i < n$. We write $first(\pi)$ resp. $last(\pi)$ for the first resp. last state of π, i.e., if π is as above then $first(\pi) = s_0$ and $last(\pi) = s_n$. Infinite paths are defined accordingly. A path is said to be maximal if it is either infinite or finite and ends in a trap.

A *strategy* (a.k.a. scheduler, policy, adversary) for $\mathcal{M}$ is a function that assigns to each non-maximal finite path π a distribution $\mathfrak{S}(\pi) \in \mathrm{Distr}(Act(last(\pi)))$. $\mathfrak{S}$ is called *deterministic* if for each non-maximal path π there is a unique action α with $\mathfrak{S}(\pi)(\alpha) = 1$, and *memoryless* if $\mathfrak{S}$'s decision only depends on the last state of the input path. *Finite-memory strategies* are those strategies that can be realized by a finite-state machine.

Given a strategy $\mathfrak{S}$ and a state $s \in S$, we write $\Pr^{\mathfrak{S}}_{\mathcal{M},s}$ or briefly $\Pr^{\mathfrak{S}}_{s}$ for the probability measure on (measurable) sets of maximal paths starting in s induced by the Markov chains obtained by unfolding $\mathcal{M}$ from s into a tree following $\mathfrak{S}$'s decisions. Note that the Markov chains of finite-memory strategies have a regular structure and can be collapsed into finite-state Markov chains. If φ is a measurable path property then we simply write $\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi)$ for the probability measure of all $\mathfrak{S}$-paths from s satisfying φ. For a worst- or best-case analysis one ranges over all strategies (i.e., all possible resolutions of the nondeterminism) and considers the extremal probabilities for satisfying φ:

$$\Pr^{\sup}_{\mathcal{M},s}(\varphi) \;=\; \sup_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi) \quad \text{and} \quad \Pr^{\inf}_{\mathcal{M},s}(\varphi) \;=\; \inf_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi)\ .$$

In case the supremum resp. infimum is met, we write $\Pr^{\max}_{\mathcal{M},s}(\varphi)$ resp. $\Pr^{\min}_{\mathcal{M},s}(\varphi)$. This, e.g., applies if φ is an ω-regular property, in which case even optimal finite-memory strategies exist.

Plain MDPs can be extended by several additional features such as a declaration of the initial states, state labels for temporal-logic specifications or weight functions for reasoning about cost and reward constraints. A *weighted MDP* is an MDP with a weight function $wgt\colon S \times Act \to \mathbb{Q}$ that assigns rational weights to all state-action pairs. The *accumulated weight* of a finite path π is denoted by $wgt(\pi)$ and is defined as the sum of weights of its state-action pairs, i.e.,

$$wgt(s_0\alpha_0 s_1\alpha_1 \ldots \alpha_{n-1}s_n) \;=\; wgt(s_i,\alpha_0) + wgt(s_1,\alpha_1) + \ldots + wgt(s_{n-1},\alpha_{n-1}) \;.$$

Given an infinite path $\varsigma = s_0\alpha_0 s_1\alpha_1 \ldots$, the upper resp. lower mean payoff is defined by

$$\overline{\mathrm{MP}}(\varsigma) = \limsup_{n\to\infty} \frac{1}{n} wgt(s_0\alpha_0 \ldots \alpha_{n-1}s_n) \quad \text{and} \quad \underline{\mathrm{MP}}(\varsigma) = \liminf_{n\to\infty} \frac{1}{n} wgt(s_0\alpha_0 \ldots \alpha_{n-1}s_n) \;.$$

It is well known that if $\mathcal{M}$ is a strongly connected Markov chain, then

$$\overline{\mathrm{MP}}(\varsigma) \;=\; \underline{\mathrm{MP}}(\varsigma) \;=\; \sum_{s\in S} \theta(s)\cdot wgt(s)$$

for almost all infinite paths ς. Here, $\theta(s)$ denotes the long-run frequency of s and $wgt(s)$ denotes the weight of the unique state-action pair (s,α) in $\mathcal{M}$. The minimal and maximal expected mean payoff in MDPs can be achieved by deterministic memoryless strategies, both computable in polynomial time using linear-programming (LP) techniques. The key idea for strongly connected MDPs is to use one variable $y_{s,\alpha}$ for the long-run frequencies of the state-action pairs (s,α) under optimal memoryless randomized strategies. This approach has been extended for general MDPs by using additional variables for the frequencies of the state-action pairs in the transient part [85]. Alternatively, one can first determine the maximal end components of the given MDP, compute the maximal or minimal expected mean payoff for them using LP techniques for strongly connected MDPs and finally compute the maximal or minimal expected mean payoff in $\mathcal{M}$. This is done by computing the maximal resp. minimal expected accumulated weight in an MDP that results from $\mathcal{M}$ by collapsing each maximal end component $\mathcal{E}$ into a single state and adding a deterministic transition from $\mathcal{E}$ to a fresh goal state whose weight is the extremal expected mean payoff in $\mathcal{E}$.

2.1 Bellman equations and the classical stochastic shortest-path problem

The *stochastic shortest-path problem* (SSP) is an optimization problem that aims to find a policy minimizing the expected costs until reaching a target. In this context, we consider a weighted MDP $\mathcal{M} = (S, Act, P, wgt)$ with a distinguished target state $goal \in S$ such that $\Pr^{\max}_{\mathcal{M},s}(\Diamond goal) = 1$ for all states $s \in S$. The task of the SSP is to determine a proper strategy that minimizes the expected accumulated weight until reaching the goal state from all states s where the minimum is taken over all proper strategies. Here, a strategy $\mathfrak{S}$ is called *proper* in case $\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\Diamond goal) = 1$ for all states s.

Theorem 2.1 ([5, 21, 55]). *SSP is solvable in polynomial time.*

Let us briefly sketch how to solve the SSP in polynomial time. Given a proper strategy $\mathfrak{S}$ and state $s \in S$, let $\mathbb{E}^{\mathfrak{S}}_{\mathcal{M},s}(\Diamond\!\!\!\!-\, goal)$ denote the expectation of the random variable $\Diamond\!\!\!\!-\, goal$ that assigns to each path ending in state $goal$ its accumulated weight. Let $x_s = \mathbb{E}^{\inf}_{\mathcal{M},s}(\Diamond\!\!\!\!-\, goal)$ denote the infimum over all proper strategies. For simplicity, let us suppose that $goal$ is a trap that is accessible from all states which ensures the existence of proper strategies. Then, the vector $(x_s)_{s\in S}$ satisfies the Bellman equations given by $x_{goal} = 0$ and

$$x_s \;=\; \min\Big\{\; wgt(s,\alpha) + \sum_{t\in S} P(s,\alpha,t)\cdot x_t \;:\; \alpha \in Act(s) \;\Big\}$$

for all states $s \in S \setminus \{goal\}$. The classical setting of [21] assumes that for each improper strategy $\mathfrak{S}$ there is at least one state $s \in S$ such that the expected total weight from s under $\mathfrak{S}$ is $+\infty$. We refer to the latter as the *SSP-assumption*. This assumption ensures that the fixed-point operator given by the Bellman equations is a contracting map in a Banach space and thus has a unique solution, obtainable as the greatest solution of the following linear constraints: $x_{goal} = 0$ and

$$x_s \;\leqslant\; wgt(s,\alpha) + \sum_{t \in S} P(s,\alpha,t) \cdot x_t \qquad \text{for all } s \in S \setminus \{goal\} \text{ and } \alpha \in Act(s).$$

This yields the polynomial-time solvability for MDPs satisfying the SSP-assumption. In [55], the cases of MDPs where either all weights are non-negative or all weights are non-positive have been considered. Furthermore, [55] presented polynomial-time algorithms to check the finiteness of the values x_s in an such an MDP $\mathcal{M}$, and if so, to transform $\mathcal{M}$ into an equivalent MDP $\mathcal{M}'$ satisfying the SSP-assumption. More recently, [5] shows how to solve the SSP for the general case[1]. Speaking roughly, while [55] ensures the SSP-assumption in MDPs with non-positive weights by collapsing end components consisting of state-action pairs (s,α) with $wgt(s,\alpha) = 0$ into a single state, [5] generalizes this approach by successively flattening end components where the accumulated weight of all cycles is 0 (called 0-ECs) using the so-called *spider construction*. The idea of the latter is to exploit the fact that for each pair (s,t) of states in a 0-EC $\mathcal{E}$ there exists a value $w(s,t) \in \mathbb{Q}$ such that the accumulated weight of all paths from s to t inside $\mathcal{E}$ have weight $w(s,t)$. In this case we can simply pick an arbitrary reference state s_0 in $\mathcal{E}$, discard all state-action pairs $(s,\alpha) \in \mathcal{E}$ and introduce deterministic transitions from each state $s \neq s_0$ to s_0 with weight $w(s,s_0)$. Finally, we replace all state-action pairs (s,β) of $\mathcal{M}$ where s is a state of $\mathcal{E}$ different from s_0 and $(s,\beta) \notin \mathcal{E}$ with the new state-action pair (s_0,β_s) where $wgt(s_0,\beta_s) = w(s_0,s) + wgt(s,\beta)$. This yields a new MDP with the same state space and the same minimal expected accumulated weight until reaching *goal* but fewer state-action pairs. The procedure can be repeated until either an end component $\mathcal{E}$ with negative minimal expected mean payoff has been found, in which case $x_s = -\infty$ for all states in $\mathcal{E}$, or the minimal expected mean payoff of all end components is positive, in which case the generated MDP satisfies the SSP-assumption.

2.2 Model checking MDPs against temporal-logic specifications

Probabilistic computation-tree logic (PCTL) [22, 78] has been introduced as a variant of CTL where the CTL path quantifiers "there is a path satisfying φ" and "all paths satisfy φ" are replaced with a probability operator. Formulas with the probability operator have the form $\mathbb{P}_I(\varphi)$ where φ is a path formula and $I \subseteq [0,1]$ an interval with rational end points. Path formulas are built as in CTL using the next and until operator as basic temporal modalities and PCTL state formulas as their arguments. When interpreting PCTL formulas over an MDP $\mathcal{M}$ with state labels, the semantics of $\mathbb{P}_I(\varphi)$ is the set of all states s in $\mathcal{M}$ such that $\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi) \in I$ for all strategies $\mathfrak{S}$. The task of the PCTL model-checking problem is to decide whether a given initial state of an MDP $\mathcal{M}$ satisfies a given PCTL state formula Φ.

Theorem 2.2 ([22, 78]). *The PCTL model-checking problem for MDPs (and Markov chains) is in P.*

The standard PCTL model-checking algorithm computes the satisfaction sets of all subformulas of Φ in a bottom-up manner ("inner subformulas first") and finally checks whether the distinguished initial state belongs to the satisfaction set of Φ. Let us consider the PCTL formula $\mathbb{P}_I(\Psi_1 \mathsf{U} \Psi_2)$ where Ψ_1, Ψ_2 are state formulas (whose satisfaction sets have been computed before and thus can be treated as sets of states) and U stands for the standard until operator. Then, the satisfaction set of $\mathbb{P}_I(\Psi_1 \mathsf{U} \Psi_2)$

[1] More precisely, [5] considers MDPs with integer weight functions. However, the approach of [5] is also applicable for rational weight functions by multiplying all weights with the least common multiple of the denominators of the weights.

is obtained by a polynomial-time graph analysis to determine all states s where $\Pr^{\widetilde{\sigma}}_{\mathcal{M},s}(\Psi_1 \mathsf{U} \Psi_2)$ equals 0 or 1 and then computing the maximal resp. minimal probabilities for the path property $\Psi_1 \mathsf{U} \Psi_2$ when ranging over all strategies. Indeed, memoryless strategies maximizing or minimizing the probabilities $\Psi_1 \mathsf{U} \Psi_2$ from all states exist and can be computed by linear-programming (LP) techniques. The encoding of maximal or minimal probabilities for reachability (resp. until) properties is similar to the Bellman equations for minimal expected accumulated weights until reaching a target (see Section 2.1).

For more complex path properties specified, e.g., using linear temporal logic (LTL), the typical task of the analysis is to check whether the probability for a given initial state and a given LTL formula φ meets a threshold condition under all strategies. This corresponds to a worst- or best-case analysis depending on whether φ is a desired or undesired event.

Theorem 2.3 ([49, 115, 116]). *The LTL model-checking problem in MDPs is 2ExpTime-complete, even in the qualitative case where the task is to decide whether an LTL formula holds almost surely or with positive probability under each or some strategy.*

The standard methods to solve the LTL model-checking problem for an MDP $\mathcal{M}$ solve the induced synthesis problem that asks to determine a strategy maximizing or minimizing the probability for φ from a given initial state. Such strategies can be obtained by first generating a deterministic automaton $\mathcal{A}$ for φ, constructing a product-MDP $\mathcal{M} \times \mathcal{A}$ and then computing a memoryless strategy for $\mathcal{M} \times \mathcal{A}$ that maximizes or minimizes the probability for reaching an end component that meets $\mathcal{A}$'s acceptance condition. As the translation of LTL formulas into deterministic automata can cause a doubly exponential blow-up, this approach yields a double exponential upper bound for the LTL model-checking problem in MDPs.

More advanced techniques for end components and several types of long-run properties, including long-run cost-utility ratios, have been proposed in [54] and [117].

2.3 Other strategy-synthesis problems

Besides refinements of strategy-synthesis algorithms with SSP-style expectation objectives or temporal-logic specifications, several other strategy-synthesis problems have been addressed in the past decade. We provide here a brief summary of a selection of such research directions.

2.3.1 Multi-ojective strategy synthesis. While the classical synthesis problems for MDPs ask to find a strategy that minimizes or maximizes the probability for a temporal property or the expectation of accumulated or discounted costs, the multi-objective synthesis problems aim to find a strategy that satisfies multiple probability and/or expectation constraints, possibly with some optimization criterion under all those strategies. For example, a typical instance of a multi-objective synthesis problem might ask to find a strategy that maximizes the expected utility subject to the requirement that the consumed energy stays below a given threshold with probability at least 0.99.

Pareto realizability for MDPs with multiple discounted reward objectives has been studied in [41]. These results have been generalized in [38] for a more expressible class of models that allow to specify discount factors depending on states and/or objectives.

In [63], a linear-programming (LP) approach to solve the synthesis problem for MDPs with multiple probabilistic reachability constraints has been proposed. The crucial observation is that randomized memoryless strategies are sufficient for this task and one can use LP techniques to check the existence of such strategies and to compute them using linear constraints with variables encoding the frequencies of the state-action pairs under optimal strategies. Analogous LP techniques have been proposed in [67, 92] to treat combinations of constraints on expected accumulated costs and probability constraints for ω-regular path properties. These techniques have been employed

for the compositional assume-guarantee verification for MDPs. Algorithms for multi-objective cost-bounded reachability conditions in MDPs are provided in [79, 106].

MDPs with multiple mean-payoff objectives in terms of expectation or probability constraints have been considered in [25]. For multiple expectation constraints, [25] proposes polynomial-time algorithms to check feasibility and to compute optimal 2-memory stochastic-update strategies. The treatment of multiple probabilistic constraints is more difficult as optimal strategies might require infinite memory. However, ε-optimal strategies are shown to be computable in polynomial time.

The "beyond worst-case" synthesis approach of [30] addresses the problem to find a strategy that is optimal with respect to an expected behavior, while satisfying a path property along all its paths. More precisely, [30] considers the SSP problem in combination with strict guarantees for a mean-payoff condition and presents a pseudo-polynomial algorithm for it. The decision problem is shown to be in NP $\cap$ coNP and at least as hard as (non-stochastic) two-player mean-payoff games.

2.3.2 Energy conditions. Energy MDPs are weighted MDPs where the integer weight function formalizes the change of an energy budget to model, e.g., the power level of a battery. Let $\mathcal{M}$ be an energy MDP with weight function $wgt\colon S\times Act \to \mathbb{Z}$. A path π in $\mathcal{M}$ fulfills the *energy objective* w.r.t. a given an initial energy budget $e \in \mathbb{N}$ when for all prefixes ρ of π we have $wgt(\rho) \geq 0$. Intuitively, an energy objective w.r.t. e stands for the requirement that the system never runs out of energy provided by a battery of capacity e. The energy objective has been introduced for quantitative models as a safety-like condition that requires the accumulated weight (e.g., available energy) to stay continuously above a given threshold [24]. Using parity conditions encoding ω-regular side constraints enables reasoning about important system properties under the energy objective:

Theorem 2.4 ([36, 100]). *The problem of deciding whether there exists an initial credit e and a strategy that achieves the energy objective w.r.t. e and a given parity condition is in NP $\cap$ coNP.*

Several related problems have been considered for energy MDPs. In [29], energy MDPs with an additional non-negative weight function (standing for the payoff achieved in some transition) have been considered, synthesizing strategies that ensure the energy condition and maximizing the expected mean payoff. Closely related to energy MDPs are one-counter MDPs with boundary [26, 27, 64], which can be seen as weighted MDPs where all weights are in $\{-1, 0, +1\}$ and terminate as soon as the counter value is 0. The termination problem asks whether there is a strategy that guarantees almost-sure termination (i.e., reaching a configuration where control is in a final location and the counter has value 0). An exponential-time algorithm for deciding the termination problem in one-counter MDPs has been presented in [26], also showing PSpace-hardness of this problem. In contrast, for weighted MDPs that do not have a lower bound on the counter values (accumulated weights), the analogous problem of almost-sure reaching a target along paths where the accumulated weight is bounded from below (or above) by a given constant under some strategy is in NP $\cap$ coNP and at least as hard as two-player mean-payoff games [5].

2.3.3 Quantiles and conditionals. A well-established concept for the synthesis of strategies in MDPs that optimize the cost-utility tradeoff relies on quantiles (see, e.g., [8, 9, 114]). Let $\mathcal{M} = (S, Act, P, wgt)$ be a weighted MDP with non-negative weight function $wgt\colon S \times Act \to \mathbb{N}$, a state $s \in S$, $p \in [0,1]\cap\mathbb{Q}$ a probability threshold, and $\trianglerighteq \in \{\geqslant, >\}$ a binary relation. In the sequel, we refer to non-negative weights as costs. Then, *quantiles* w.r.t. probabilistic constraints for until properties on state sets $A, B \subseteq S$ with upper or lower cost bounds and lower probability bounds are provided

through

universal quantiles:

$$\min \left\{ c \in \mathbb{N} \,:\, \Pr^{\min}_{\mathcal{M},s}(A \,\mathsf{U}^{\leqslant c} B) \trianglerighteq p \right\}$$
$$\max \left\{ c \in \mathbb{N} \,:\, \Pr^{\min}_{\mathcal{M},s}(A \,\mathsf{U}_{\geqslant c} B) \trianglerighteq p \right\}$$

existential quantiles:

$$\min \left\{ c \in \mathbb{N} \,:\, \Pr^{\max}_{\mathcal{M},s}(A \,\mathsf{U}^{\leqslant c} B) \trianglerighteq p \right\}$$
$$\max \left\{ c \in \mathbb{N} \,:\, \Pr^{\max}_{\mathcal{M},s}(A \,\mathsf{U}_{\geqslant c} B) \trianglerighteq p \right\}$$

For instance, the existential quantile $\min\{ c \in \mathbb{N} : \Pr^{\max}_{\mathcal{M},s}(A \,\mathsf{U}^{\leqslant c} B) > p \}$ asks for the minimal cost c that need to be spend to guarantee reaching B through A-states with probability exceeding p. Encoding a second non-negative integer weight function into the state space of $\mathcal{M}$ and formalizing a notion of utility enables to reason about the cost-utility tradeoff by only regarding states in B that exceed a given utility threshold. While qualitative quantiles, i.e., $\trianglerighteq p$ is either "> 0" or "$= 1$", are computable in polynomial time using variants of Dijkstra's shortest-path algorithm, pseudo-polynomial algorithms for computing cost-bounded reachability probabilities and related quantiles have been presented in [6, 114]. Indeed, no algorithms with better complexity can be expected:

Theorem 2.5 ([72]). *Deciding whether there is a strategy in an MDP $\mathcal{M}$ such that the probability of cost-bounded properties exceeds a given probability threshold is PSpace-hard, and PSpace-complete for the case where $\mathcal{M}$ is acyclic.*

Conditional probabilities and expectations are an important concept, e.g., for a quantitative analysis of stochastic systems with rare event scenarios. Given an MDP $\mathcal{M} = (S, Act, P)$, a state $s \in S$, and path properties φ and ψ, we consider the problem of computing

$$\Pr^{\max}_{\mathcal{M},s}(\varphi \mid \psi) \quad \stackrel{\text{def}}{=} \quad \max_{\mathfrak{S}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi \mid \psi) \quad = \quad \max_{\mathfrak{S}} \frac{\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\varphi \wedge \psi)}{\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\psi)} \,.$$

Here, φ is called the *objective* and ψ the *condition* of the conditional probability, respectively. Minimal conditional probabilities are defined analogously, but one can savely restrict to the maximizing case as $\Pr^{\min}_{\mathcal{M},s}(\varphi \mid \psi) = 1 - \Pr^{\max}_{\mathcal{M},s}(\neg\varphi \mid \psi)$. In [2], a model-checking algorithm for MDPs and PCTL formulas extended by constraints for conditional probabilities has been presented, running in exponential time when both the objective and the condition are reachability properties. Using the *reset method* proposed in [14], the synthesis of maximizing strategies for conditional probabilities with reachability objective and condition can, however, be achieved in polynomial time. The latter approach has been generalized for ω-regular properties, implemented in state-of-the art model checkers such as Storm [57] and Prism [99], and applied in several case studies. In [15], an exponential-time algorithm for computing the maximal conditional expectation and a corresponding optimal strategies has been established. Recently, concepts for conditional probabilities and quantiles have been combined in computing conditional-value-of-risk for mean-payoff and reachability objectives in Markov chains and MDPs [89].

2.3.4 Robust systems: optimization under resilience constraints. In the literature, there are many facets of describing the ability of some system to react on errors and environmental perturbations (see, e.g., surveys [4, 110]). We here concentrate on a few recently established aspects concerning synthesis problems to illustrate that this area is an active field of research, not limited to probabilistic systems but also important for non-probablistic ones. The general idea behind most of the synthesis problems is to determine strategy decisions that increase resiliency of the system. Following [110], resilience disciplines capture classical notions such as robustness, survivability, and fault tolerance. Robustness is a control-theoretic property relating the operation of a system to perturbations of its input. The task in robust reactive synthesis is to generate a controller guaranteeing an ω-regular property under environment assumptions that might be violated temporarily (see, e.g., [23, 62]).

In particular, [23] considered quantitative notions of robustness formalized using mean payoff conditions. Robustness in (non-probabilistic) game structures has been considered, e.g., in [81], establishing a framework for synthesizing a strategy for one player in a safety game that maximizes the resilience under multiple (but within a fixed bound) disturbances. Similarly, resilient strategies that are optimal w.r.t. the number of disturbances covered were synthesized in [102], also using a game-theoretic approach that is not limited to safety games only. In [98] a formal definition of robustness w.r.t. ω-regular properties using distance functions and an algorithm to synthesize robust strategies has been presented.

The synthesis of resilient strategies has not only been considered for non-probabilistic systems as in the aforementioned approaches, but also in the probabilistic setting, where probabilities occur, e.g., when modeling environmental disturbances or components such as sensors or within the synthesis of randomized resilient strategies. In [59], a permissive controller synthesis framework has been introduced, generating multi-strategies to represent alternatives for controller decisions that minimize the penalties to be paid for "disallowed" actions. Their formalization relies on stochastic two-player games with probabilistic reachability constraints (via PCTL-like expressions) or constraints on the expected accumulated reward. The underlying decision problem is shown to be NP-hard and solvable using an incremental approach with mixed integer linear programming. Recently, resilient control strategies have been defined for MDP models [11]. The authors call a strategy resilient when it ensures the recovery of the system under a given cost and probability constraint. They show that deciding whether there exists a resilient strategy is PSpace-hard and can be done in pseudo-polynomial time. The synthesis of a resilient strategy that optimizes the long-run average reward under all resilient strategies is shown to be computable in pseudo-polynomial time using an LP-based approach.

3 FEATURE-ORIENTED SYNTHESIS

In feature-oriented system development (see, e.g., [3] for an overview), a *feature* describes a user-visible aspect or characteristics that may be present or absent in a system [86]. Features are most prominently used to model variability in (software) product lines [46], i.e., families of system variants that rely on a common code base and differ in the selection of features. Several extensions of the classical Boolean setting for features have been proposed in the literature. For instance, *multi-features* [52] are features that may not only be absent or present in a variant but can appear multiple times, features with *attributes* [48] are features that depend on (numerical) parameters, and *dynamic features* [71] are features that can be changed during run-time of the system. The classical setting for dynamic features amounts of simply activating or deactivating the feature, but also feature attributes or cardinalities might be subject of run-time changes for modeling adaptations within the system family. Beside others, these extensions enable feature-oriented concepts to be used also for a wide area of applications, e.g., to model and analyze adaptive and parametric systems.

The standard formalism for constructing variants in feature-oriented systems is provided by *superimposition* [87], also apparent in delta-oriented programming [108]. For each feature it is specified which other features are required to be present and absent and how including the feature in a variant changes the behaviors of the system, i.e., what behaviors are removed, added, or modified. Opposed to the superimposition concept, *featured transition systems* provide a monolithic model to encode all behaviors of variants into a single model [45]. In [60], a compositional formalism for dynamic feature-oriented systems has been introduced. The information about active features in a variant and their run-time switches is represented by an automata-based component called *feature controller*. Behaviors of features are encapsulated in separated components, composed with the feature controller using standard parallel-composition operations. These concepts naturally extend to probabilistic feature-oriented systems [60] where feature components and controller

are given by MDPs. In [44], the probabilistic framework of [60] has been implemented in the tool ProFeat, extending the input language of the prominent probabilistic model checker Prism by feature-oriented concepts and supporting the family-based analysis of feature-oriented systems.

Family-based analysis. For the analysis of feature-oriented systems, one mainly distinguishes between two approaches: *one-by-one* and *all-in-one analysis*. Within a one-by-one analysis each variant is analyzed separately, while an all-in-one analysis uses a *family model* that encodes all variants in a single model and deduces the results for each variant from analyzing the family model in a single run. As the number of variants might be exponential in the number of features, a one-by-one analysis easily becomes infeasible for complex feature-oriented systems. Exploiting the fact that variants share behaviors of common features, symbolic representations of the family model might tackle the exponential blow-up apparent in one-by-one analysis approaches. As a consequence, all-in-one approaches are usually superior to one-by-one approaches, witnessed by several case studies in the literature (see, e.g., the survey [113]).

3.1 Variant selection and featured strategy synthesis

The analysis of feature-oriented system usually asks for those variants that fulfill a given property, might it be functional (e.g., whether a safety property is fulfilled or not) or non-functional (e.g., whether a PCTL property is satisfied). In the context of probabilistic feature-oriented systems the task can be further extended to select *optimal variants*, e.g., asking for those variants maximizing the probability to reach a target or minimizing the expected energy consumption. Although not stated explicitly, the approach of [68] can be used to select optimal variants following a given optimization criterion, using the parametric engine of Prism to obtain rational functions for probabilities and expectations. This applies also to case studies following this approach, e.g., [93, 107]. Using the approach by [60], the selection of energy-optimal variants in heterogeneous tiled architectures were determined in [16].

When dynamic features are involved in the system under consideration, another dimension regarding the selection of optimal variants arises. Then, also the synthesis of an optimal strategy how to activate and deactivate features during run-time is of interest. This synthesis problem has been considered in [61] and [60] for energy-aware server systems, extended towards synthesizing strategies that are optimal w.r.t. the energy-utility tradeoff expressed through energy-utility quantiles in [88].

3.2 Feature-oriented parameter synthesis

We now present another aspect in feature-oriented synthesis that has not yet been considered in the literature. Systems those behaviors depend on configurable parameters might be elegantly modeled through feature-oriented concepts including feature attributes for representing the system parameters. The arising system family then spans an optimization space over these parameters and one can exploit family-based analysis methods to synthesize optimal parameters. We exemplify the approach by synthesizing energy-utility optimal parameters for the energy-aware network device eBond [76].

eBond. In the recent past, the focus of data-center design has seen a steady move from a purely performance-optimizing approach to an energy-aware approach due to cost, environmental, and infrastructural issues. For existing data-centers that do not yet include energy-aware hardware, [76] proposed a technology to adapt the usage of heterogeneous network interface cards (NICs). The test bed on which the authors evaluated their approach comprised a slow but low-energy 1 Gbit/s NIC (consuming between 1.41 W and 1.77 W) and a fast but high-energy 10 Gbit/s NIC (consuming between 7.86 W and 8.06 W). The eBond network scheduling algorithm has three parameters:

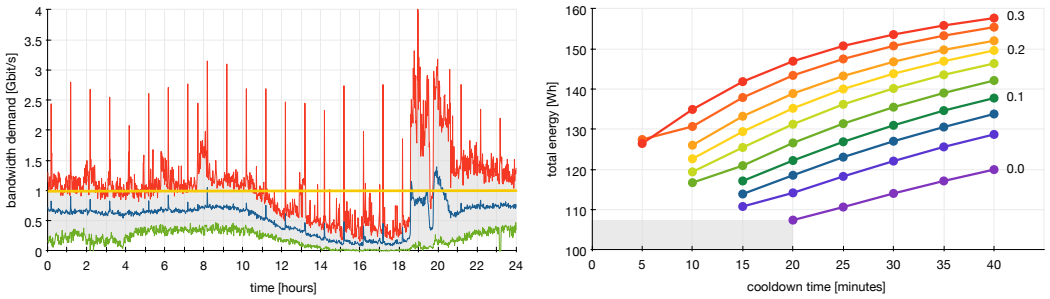


Fig. 1. Bandwidth distribution (left) and solution space for $p = 0.95$ (right)

interval Δ (in minutes), cooldown x (in minutes), and predictor $y \in [0, 1] \cap \mathbb{Q}$. After each Δ minutes, the requested average bandwidth b (in Gbit/s) within the last Δ minutes and the current cooldown timer t is taken into account for deciding whether to switch the active NIC. In case $b \cdot (1 + y) > 1$ Gbit/s, the cooldown timer is reset to x and the 10 Gbit NIC is activated. Otherwise, the cooldown timer is decreased by Δ and the 1 Gbit/s NIC is activated in case $t \leq 0$. The authors considered three variants of the algorithm: *balanced* ($x = 30$ minutes and $y = 0.1$), *high savings* ($x = 0$ minutes and $y = 0.1$), and *aggressive* ($x = 0$ minutes and $y = 0$). Using a simulation-based approach on real-word server data logs, they demonstrated that significant energy-savings can be achieved with these variants. The potential of saving energy mainly stems from the different bandwidth demands during day and night time. However, energy savings come at the cost of package delays, occurring when the bandwidth demands exceed 1 Gbit/s while on the slow NIC. Hence, there is a tradeoff between saving energy while guaranteeing utility in terms of service quality of the data center.

Feature-oriented model of EBOND. To model the EBOND protocol as a feature-oriented system, we rely on the framework presented in [44, 60]. The static `parameter` feature with feature attributes x (cooldown) and y (predictor) set the decision-making parameters. Two dynamic features `1Gbit` and `10Gbit` model the NICs, those switches are maintained by a deterministic feature controller component according to the EBOND protocol described above, making decisions depending on the feature attributes x and y of the `parameter` feature. The power measurements for both NICs from [76] are discretized with a granularity of 25 Gbit/s and included in the respective MDP-model of the features as non-negative weight function. A static `environment` feature with the feature attribute Δ (interval) encapsulates the average bandwidth requirements in time steps of $\Delta = 5$ (minutes), obtained from statistical analysis of real-world server data logs: We exploit the known shift between day and night, convoluting the 42 days ubuntu-release server log from [76] into a probability distribution over one single day. As for the power measurements, a granularity of 25 Gbit/s is used. Figure 1 illustrates the bandwidth convolution, depicting the minimal (green), average (blue), and maximal (red) bandwidth demands occurring in the used server logs. We also highlighted the critical threshold at 1 Gbit/s for switching from the slow low-energy NIC to the fast high-energy NIC. Note that in the related case studies of [60, 61, 88] the bandwidth demands were modeled without using statistical data over real-word server logs.

Family-based synthesis of energy-utility optimal EBOND *parameters.* While the simulation-based approach of [76] allowed for the estimation of the energy consumption and the level of utility provided by the EBOND protocol for given parameters, it is not capable to reason about the energy-utility tradeoff and to synthesize energy-utility optimal parameters. We now show how this can be achieved using probabilistic model-checking techniques applied on the feature-oriented model

of eBond. To synthesize parameters x and y that provide optimal energy-utility tradeoff, we consider the attributes of the `parameter` feature, where the cooldown parameter x ranges from 0 to 40 minutes in Δ-steps and the predictor parameter y ranges from 0 to 0.3 in $\frac{1}{30}$-steps. To this end, the eBond-protocol family comprises 90 variants. We considered an energy-utility quantile (see Section 2.3.3) over one day, i.e., the set of target states end_of_day is determined by those states where the time bound of 288 Δ-steps is reached. As proposed in [6], the utility measure u is encoded into the state space and corresponds to the number of Δ-steps where no package delay occurred. The utility constraint utility_goal is given by the set of states where $u \geq 276$, i.e., in at most 12 time slots of $\Delta = 5$ minutes, a package delay occurred. Let us define a goal function $q\colon \{0, 5, \ldots, 40\} \times \{0, 0.0\overline{3}, 0.0\overline{6}, \ldots, 0.3\} \times [0,1] \cap \mathbb{Q} \to \mathbb{Q}$ by the following energy-utility quantile values w.r.t. a probability bound p:

$$q(x,y,p) \;\mapsto\; \min\left\{\, e \in \mathbb{Q} : \Pr_{x,y}\big(\Diamond^{\leqslant e}(\text{end\_of\_day} \wedge \text{utility\_goal})\big) > p \,\right\} .$$

Here, $\Pr_{x,y}(\cdot)$ denotes the probability measure of the variant with parameters x and y. Given $p \in [0,1] \cap \mathbb{Q}$, the eBond parameter-synthesis problem now amounts to find a parameter combination that minimizes $q(\cdot,\cdot,p)$, i.e., when the probability threshold is provided through the service contract on the data center the eBond technology is applied on, we ask for the parameter combination that guarantees the energy-utility tradeoff with a minimal amount of energy. We solved the eBond parameter-synthesis problem for the probability threshold $p = 1$, as the algorithm behind quantitative quantile computations [6] is conceptional able to report on intermediate probability bounds and their quantile values. On the right of Figure 1, the results of the family-based analysis are shown for $p = 0.95$, where each plot corresponds to a fixed value of the predictor parameter y. Following our optimization criterion, one can easily determine that the parameter combination $x = 20$ minutes and $y = 0$ is optimal, guaranteeing the required level of service quality with a daily energy investment of 107.43 Wh. This illustrates that the predictor parameter y does not have a big impact on the utility-level guarantees and increasing the cooldown time x is more effective.

Table 1. Statistics on energy-optimal eBond parameter synthesis

analysis method	number of states	number of nodes	analysis time [s]
one-by-one	80'137'366	7'665'989	66'425.137
all-in-one	80'137'366	221'125	15'371.919

We carried out[2] both, a one-by-one and an all-in-one analysis of the eBond feature-oriented system to solve the eBond parameter-sythesis problem using the semi-symbolic Sparse engine of the model checker Prism [91] with variable reordering enabled [88]. Table 1 reports on the statistics of the experiments, indicating the number of states of the resulting model, the number of MTBDD nodes[3] used for the symbolic representation of the model, and the timings of the analysis. In case of the one-by-one approach, the values correspond to the sum of the characteristics of all analysis runs on single variants. In the number of nodes one can observe that the family model indeed exploits the shared behaviors of common features, e.g., the comparably sophisticated `environment` feature. The analysis time is more than four times shorter in case of the all-in-one analysis.

[2]Hardware setup: Intel Xeon E5-2680@2.70GHz, 128 GB RAM; Turbo Boost and HT enabled; Debian GNU/Linux 9.1
[3]Prism uses multi-terminal binary decision diagrams (MTBDDs) [80] for its symbolic engines.

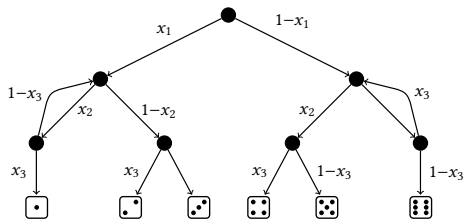
4 SYNTHESIS OF PROBABILITY PARAMETERS

Results of the formal analysis and synthesis with Markovian models and quantitative specifications crucially depend on the concrete transition probabilities. Even small perturbations of the probability values can affect the analysis or synthesis results. This can be problematic in cases where only estimates of the transition probabilities are available. This, for instances, applies to cases where probability values in the models are derived using statistical or learning methods.

This motivated the introduction of Markov chains where intervals are attached to the transitions rather than concrete transition probabilities [19, 31, 42, 58, 84, 109]. Two semantics have been introduced for interval-valued Markov chains. The *uncertain semantics* treats interval-valued Markov chains as families of Markov chains with the same state space and transition probabilities in the intervals. The *nondeterministic semantics* considers interval-valued Markov chains as MDPs where the concrete probability values are chosen nondeterministically.

Parametric Markov chains can be seen as a generalization of interval-valued models with the uncertain semantics. Instead of intervals of potential probability values, they use polynomial functions over a fixed set of parameters to specify the transition probabilities [53, 75, 94]. Such parametric models can be seen to define a family of concrete probabilistic models that arise by plugging in concrete values for the parameters.

Formally, a parametric Markov chain with parameters $x_1, \ldots, x_k$ is a tuple $\mathcal{M}[x_1, \ldots, x_k] = (S, E, \Upsilon)$ where (S, E) is a finite directed graph, i.e., S is a finite state space and $E \subseteq S \times S$ specifies the transitions, $\Upsilon \colon E \to \mathbb{Q}[x_1, \ldots, x_k]$ a function that assigns to each transition $s \to s'$ a polynomial over $x_1, \ldots, x_k$ with rational coeffients. A parameter valuation $\xi \colon \{x_1, \ldots, x_k\} \to \mathbb{R}$ assigning real numbers to the parameters is said to be *admissible* if $\sum_{s' \in E(s)} \Upsilon(s, s')(\xi_1, \ldots, \xi_k) = 1$ for each non-trap state $s \in S$ where $E(s) = \{s' \in S : (s, s') \in E\}$. The (concrete) Markov chain given by an admissible parameter valuation ξ is $\mathcal{M}_\xi = (S, P_\xi)$ where $P_\xi(s, s') = \Upsilon(s, s')(\xi_1, \ldots, \xi_k)$ if $(s, s') \in E$ and $P_\xi(s, s') = 0$ if $(s, s') \notin E$. Then, the semantics of $\mathcal{M}[x_1, \ldots, x_k]$ is the family of (concrete) Markov chains induced by the admissible parameter valuations.



probability for the outcome ⚅
(rational function)

$$\frac{-x_3x_2x_1+x_2x_1+x_3x_1+x_3x_2-x_1-x_2-x_3+1}{x_3x_2-x_3+1}$$

Fig. 2. Parametric Markov chain for the simulation of a dice by flipping biased coins

On the left of Figure 2 a parametric Markov chain is depicted, describing a parametric variant of Knuth and Yao's protocol for simulating a six-sided dice by coin-flipping.

Probabilities for reachability conditions (and even for all ω-regular properties) can be represented as the solution of a linear equation system $A \cdot p = b$ where A is a matrix with coefficients represented as polynomials derived from Υ, p is the probability vector, and b a column vector. Thus, we can view A as a matrix over the field $\mathbb{Q}(x_1, \ldots, x_k)$ of rational functions with parameters $x_1, \ldots, x_k$ and then apply Gaussian elimination (or the related state-elimination approach known for generating regular expressions for finite automata) to compute the solution vector p. This approach has been first proposed in [53, 94] and later refined using computer-algebra tools to simplify the rational functions

obtained in intermediate steps [75] or using decompositions into strongly connected components and factorization techniques for polynomials [83]. These techniques have been implemented in the tools PARAM [74], its reimplementation in PRISM [91], and in STORM [57]. More recently, it has been observed that the expensive computations of greatest common devisors of polynomials can be avoided using one-step fraction-free Gaussian elimination [82]. The latter yields:

THEOREM 4.1 ([82]). *The rational functions for reachability probabilities in parametric Markov chains are computable in time $O(\text{poly}(n,d)^k)$ where n is the number of states, d the maximal degree of the polynomials in Υ and k the number of parameters.*

The same holds for expected accumulated weights or the expected mean payoff. Thus, in the univariate case k=1 (or for any fixed number k of parameters), the rational functions are computable in polynomial time.

The applications of parametric Markov chains and the algorithms to compute rational functions for reachability probabilities or expected values are manyfold. These include the analysis of systems with unknown transition probabilities where the task is a family-based analysis to provide guarantees for all admissible parameter valuations, but also the treatment of systems with tiny transition probabilities, which, e.g., applies often to models that capture the effect of very exceptional errors and where standard analysis techniques for the analysis fail due to numerical problems.

The parameter-synthesis problem goes one step further and asks to find parameter valuations where a set probability or expectation constraints holds, possibly in combination with other side constraints. For a simple example, we regard the Markov chain shown in Figure 2. While for a fair coin, $x_1 = x_2 = x_3$, the result of the outcome "six" is 1/6, one might ask how to manipulate the coins such that the probability for "six" is 1/2. Assuming the same coin is used in all rounds (i.e., $x_1 = x_2 = x_3$), the task is to find a value p such that $(-p^3 + 3p^2 - 3p + 1)/(p^2 - p + 1) = 1/2$. With Newton's method we obtain $p \approx 0.26102$. While this is a toy example, several important applications of the parameter-synthesis problem have been studied in the literature. Let us mention a few of them.

The use of parametric model checking for supporting decision making at run-time in adaptive software has been investigated by several authors, see, e.g., [32, 66, 112]. The idea is to precompute the rational functions for a set of quantitative properties, which can be efficiently evaluated at run-time for guiding software adaptions when the environment changes. An iterative decision-making scheme for MDPs with interval estimates for the transition probabilities has been proposed in [111]. This approach serves to tackle the trade-off between accuracy, data usage and computational overhead. It successively computes strategies that are optimal for a cost-bounded reachability condition and checks their confidence optimality. If not, the iteration returns to data sampling.

Besides supporting decisions at run-time, the parametric approach can also be very useful to find (nearly) optimal system configurations at design-time. For example, [1] employs parametric model-checking techniques to find probability distributions for a probabilistic self-stabilizing protocol that achieves minimum average recovery time, while [95] uses the parametric approach to determine (an approximation of) the optimal frequency of periodic reboots for an inter-process communication protocol of a space probe where optimality is understood with respect to the long-run availability. In [18] the model-repair problem is considered where some transition probabilities of a Markov chain are declared to be controllable and the task is to modify the controllable probabilities such that a PCTL property holds for the modified Markov chain and the costs for deriving the new Markov chain are minimal. The latter is formalized using a nonlinear cost function, which leads to a nonlinear programming problem and is shown to be related to the optimal-controller synthesis problem for discrete linear dynamical systems. An alternative model-repair approach using parametric Markov chains and greedy methods has been proposed in [104].

Beyond parametric Markov chains, several authors have addressed the parametric setting to different and more expressive stochastic models. This includes methods to synthesize parametric rate values in continuous-time Markov chains that ensure the validity of bounded reachability properties [77]. Timeout-synthesis problems for continuous-time stochastic models have been investigated, e.g., in [10, 28]. The parameter-synthesis problem for interval Markov chains with parametric lower and upper endpoints of intervals has been addressed in [17] where methods to find concrete values for the parameters such that the resulting interval-valued Markov chain maximizes the probability for a reachability condition have been proposed. Parametric approaches for MDPs have been studied in [50, 51, 73]. Another recent approach to deal with unknown transition probabilities uses online learning techniques, e.g., applied on MDP models in [90], where only the support of distributions is known in advance and where the task is to synthesize a strategy that almost surely satisfies a parity condition and is nearly optimal w.r.t. a mean-payoff objective.

5 CONCLUSION

In this article we gave a brief overview on traditional and recent directions for synthesis problems based on MDPs with manyfold application areas. The first part reports on synthesis problems of the classical type, i.e., synthesizing strategies, but with non-standard objectives. A common concept of the last two sections focused on family-based analysis that supports to determine system instances (family members) satisfying certain constraints. This article is far from being complete and there are several other directions not mentioned so far. One direction are strategy-synthesis problems for partially observable MDPs, POMDPs for short, [96, 103]. The challenge here is that strategies do not have access to the full history. Many verification problems are conceptually close to decision problems for probabilistic finite automata and therefore undecidable. This, for instance, applies to algorithmic questions for expected costs [97], but also to verification problems for infinite-horizon properties under qualitative probability thresholds [12] or for distributed strategies [69]. To escape from undecidability, one can restrict to finite-horizon properties (see, e.g., [101]) or can impose syntactic restrictions on the type of strategies. For instance, ExpTime-completeness has been established for finite-memory strategies and parity objectives and almost-sure satisfaction in POMDPs [34]. Further, the stochastic shortest-path problem for POMDPs with positive cost functions was shown to be solvable in double exponential time [33]. Decidability has also been established for restricted classes of strategies in probabilistic distributed systems [70]. Another very active research area is the strategy synthesis in stochastic two- or multi-player games (see, e.g., [20, 35, 37, 43, 47, 56]).

REFERENCES

[1] Saba Aflaki, Matthias Volk, Borzoo Bonakdarpour, Joost-Pieter Katoen, and Arne Storjohann. 2017. Automated Fine Tuning of Probabilistic Self-Stabilizing Algorithms. In *36th IEEE Symposium on Reliable Distributed Systems (SRDS)*. IEEE Computer Society, 94–103.

[2] Miguel E. Andrés. 2011. *Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems*. Ph.D. Dissertation. UB Nijmegen.

[3] Sven Apel and Christian Kästner. 2009. An Overview of Feature-Oriented Software Development. *Journal of Object Technology* 8 (2009), 49–84.

[4] Nii Attoh-Okine. 2016. *Resilience Engineering: Models and Analysis*. Cambridge University Press.

[5] Christel Baier, Nathalie Bertrand, Clemens Dubslaff, Daniel Gburek, and Ocan Sankur. 2018. Stochastic Shortest Paths and Weight-Bounded Properties in Markov Decision Processes. In *33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM, 86–94.

[6] Christel Baier, Marcus Daum, Clemens Dubslaff, Joachim Klein, and Sascha Klüppelholz. 2014. Energy-Utility Quantiles. In *6th NASA Formal Methods Symposium (NFM) (Lecture Notes in Computer Science)*, Vol. 8430. 285–299.

[7] Christel Baier, Luca de Alfaro, Vojtech Forejt, and Marta Kwiatkowska. 2018. Model Checking Probabilistic Systems. In *Handbook of Model Checking*. Springer, 963–999.

[8] Christel Baier, Clemens Dubslaff, and Sascha Klüppelholz. 2014. Trade-off Analysis Meets Probabilistic Model Checking. In *23rd Conference on Computer Science Logic and the 29th Symposium on Logic In Computer Science (CSL-LICS)*. ACM, Article 1, 1:1–1:10 pages. Invited talk.

[9] Christel Baier, Clemens Dubslaff, Sascha Klüppelholz, Marcus Daum, Joachim Klein, Steffen Märcker, and Sascha Wunderlich. 2014. Probabilistic Model Checking and Non-standard Multi-objective Reasoning. In *Fundamental Approaches to Software Engineering*, Stefania Gnesi and Arend Rensink (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16.

[10] Christel Baier, Clemens Dubslaff, Ľuboš Korenčiak, Antonín Kučera, and Vojtěch Řehák. 2017. Mean-Payoff Optimization in Continuous-Time Markov Chains with Parametric Alarms. In *14th International Conference on Quantitative Evaluation of Systems (QEST) (Lecture Notes in Computer Science)*, Vol. 10503. Springer, 190–206.

[11] Christel Baier, Clemens Dubslaff, Lubos Korenciak, Antonín Kucera, and Vojtech Rehák. 2017. Synthesis of Optimal Resilient Control Strategies. In *15th International Symposium on Automated Technology for Verification and Analysis (ATVA) (Lecture Notes in Computer Science)*, Vol. 10482. Springer, 417–434.

[12] Christel Baier, Marcus Größer, and Nathalie Bertrand. 2012. Probabilistic ω-automata. *J. ACM* 59, 1 (2012), 1:1–1:52.

[13] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

[14] Christel Baier, Joachim Klein, Sascha Klüppelholz, and Steffen Märcker. 2014. Computing Conditional Probabilities in Markovian Models Efficiently. In *20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (Lecture Notes in Computer Science)*, Vol. 8413. 515–530.

[15] Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. 2017. Maximizing the Conditional Expected Reward for Reaching the Goal. In *23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (Lecture Notes in Computer Science)*, Vol. 10206. 269–285.

[16] Christel Baier, Sascha Klüppelholz, and Sascha Wunderlich. 2017. Towards Automated Variant Selection for Heterogeneous Tiled Architectures. In *Models, Algorithms, Logics and Tools: Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday (Lecture Notes in Computer Science)*, Vol. 10460. Springer, 382–399.

[17] Anicet Bart, Benoît Delahaye, Paulin Fournier, Didier Lime, Eric Monfroy, and Charlotte Truchet. 2018. Reachability in parametric Interval Markov Chains using constraints. *Theoretical Computer Science* 747 (2018), 48–74.

[18] Ezio Bartocci, Radu Grosu, Panagiotis Katsaros, C. R. Ramakrishnan, and Scott A. Smolka. 2011. Model Repair for Probabilistic Systems. In *17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (Lecture Notes in Computer Science)*, Vol. 6605. Springer, 326–340.

[19] Michael Benedikt, Rastislav Lenhardt, and James Worrell. 2013. LTL Model Checking of Interval Markov Chains. In *19th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (LNCS)*, Vol. 7795. Springer, 32–46.

[20] Nathalie Bertrand, Blaise Genest, and Hugo Gimbert. 2017. Qualitative Determinacy and Decidability of Stochastic Games with Signals. *Journal of the ACM* 64, 5 (2017), 33:1–33:48.

[21] Dimitri P. Bertsekas and John N. Tsitsiklis. 1991. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research* 16(3) (1991), 580–595.

[22] Andrea Bianco and Luca de Alfaro. 1995. Model checking of probabilistic and non-deterministic systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS) (Lecture Notes in Computer Science)*, Vol. 1026. 499–513.

[23] Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer. 2014. Synthesizing robust systems. *Acta Informatica* 51, 3-4 (2014), 193–220.

[24] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. 2008. Infinite Runs in Weighted Timed Automata with Energy Constraints. In *6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS) (Lecture Notes in Computer Science)*, Vol. 5215. Springer, 33–47.

[25] Tomás Brázdil, Václav Brozek, Krishnendu Chatterjee, Vojtěch Forejt, and Antonín Kucera. 2014. Markov Decision Processes With Multiple Long-run Average Objectives. *Logical Methods in Computer Science* 10, 1 (2014).

[26] Tomás Brázdil, Václav Brozek, Kousha Etessami, and Antonín Kucera. 2013. Approximating the termination value of one-counter MDPs and stochastic games. *Information and Computation* 222 (2013), 121–138.

[27] Tomás Brázdil, Václav Brozek, Kousha Etessami, Antonín Kucera, and Dominik Wojtczak. 2010. One-Counter Markov Decision Processes. In *Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 863–874.

[28] Tomás Brázdil, Lubos Korenciak, Jan Krcál, Petr Novotný, and Vojtech Rehák. 2015. Optimizing Performance of Continuous-Time Stochastic Systems Using Timeout Synthesis. In *12th International Conference on Quantitative Evaluation of Systems (QEST) (Lecture Notes in Computer Science)*, Vol. 9259. Springer, 141–159.

[29] Tomás Brázdil, Antonín Kucera, and Petr Novotný. 2016. Optimizing the Expected Mean Payoff in Energy Markov Decision Processes. In *14th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, Vol. 9938. 32–49.

[30] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. 2014. Meet Your Expectations With Guarantees: Beyond Worst-Case Synthesis in Quantitative Games. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 199–213.

[31] Benoît Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. 2011. Constraint Markov Chains. *Theoretical Computer Science* 412, 34 (2011), 4373–4404.

[32] Radu Calinescu, Carlo Ghezzi, Marta Z. Kwiatkowska, and Raffaela Mirandola. 2012. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM* 55, 9 (2012), 69–77.

[33] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. 2016. Optimal cost almost-sure reachability in POMDPs. *Artificial Intelligence* 234 (2016), 26–48.

[34] Krishnendu Chatterjee, Martin Chmelik, and Mathieu Tracol. 2016. What is decidable about partially observable Markov decision processes with ω-regular objectives. *Journal of Computer and System Science* 82, 5 (2016), 878–911.

[35] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. 2013. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of Computer and System Science* 79, 5 (2013), 640–657.

[36] Krishnendu Chatterjee and Laurent Doyen. 2011. Energy and Mean-Payoff Parity Markov Decision Processes. In *36th International Symposium on Mathematical Foundations of Computer Science (MFCS) (Lecture Notes in Computer Science)*, Vol. 6907. 206–218.

[37] Krishnendu Chatterjee and Laurent Doyen. 2016. Perfect-Information Stochastic Games with Generalized Mean-Payoff Objectives. In *31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM, 247–256.

[38] Krishnendu Chatterjee, Vojtěch Forejt, and Dominik Wojtczak. 2013. Multi-objective Discounted Reward Verification in Graphs and MDPs. In *19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR) (Lecture Notes in Computer Science)*, Vol. 8312. 228–242.

[39] Krishnendu Chatterjee and Monika Henzinger. 2011. Faster and Dynamic Algorithms for Maximal End-Component Decomposition and Related Graph Problems in Probabilistic Verification. In *22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1318–1336.

[40] Krishnendu Chatterjee, Thomas A. Henzinger, Barbara Jobstmann, and Rohit Singh. 2015. Measuring and Synthesizing Systems in Probabilistic Environments. *J. ACM* 62, 1 (2015), 9:1–9:34.

[41] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. 2006. Markov Decision Processes with Multiple Objectives. In *23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS) (Lecture Notes in Computer Science)*, Vol. 3884. 325–336.

[42] Krishnendu Chatterjee, Koushik Sen, and Thomas A. Henzinger. 2008. Model-Checking omega-Regular Properties of Interval Markov Chains. In *11th Int. Conf. on Foundations of Software Science and Computational Structures (FoSSaCS) (LNCS)*, Vol. 4962. Springer, 302–317.

[43] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Aistis Simaitis. 2013. Automatic verification of competitive stochastic systems. *Formal Methods in System Design* 43, 1 (2013), 61–92.

[44] Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. 2018. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Aspects of Computing* 30, 1 (2018), 45–75.

[45] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-François Raskin. 2013. Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. *IEEE Transactions on Software Engineering* 39, 8 (2013), 1069–1089.

[46] Paul C. Clements and Linda M. Northrop. 2001. *Software Product Lines : Practices and Patterns*. Addison-Wesley Professional.

[47] Anne Condon. 1992. The Complexity of Stochastic Games. *Information and Computation* 96, 2 (1992), 203–224.

[48] Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. 2013. Beyond Boolean Product-line Model Checking: Dealing with Feature Attributes and Multi-features. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, Piscataway, NJ, USA, 472–481.

[49] Costas Courcoubetis and Mihalis Yannakakis. 1995. The Complexity of Probabilistic Verification. *J. ACM* 42, 4 (1995), 857–907.

[50] Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, Ivan Papusha, Hasan A. Poonawala, and Ufuk Topcu. 2017. Sequential Convex Programming for the Efficient Verification of Parametric MDPs. In *TACAS (2) (LNCS)*, Vol. 10206. 133–150.

[51] Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. 2018. Synthesis in pMDPs: A Tale of 1001 Parameters. In *16th International Symposium on Automated Technology for Verification and Analysis (ATVA) (Lecture Notes in Computer Science)*, Vol. 11138. Springer, 160–176.

[52] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2004. Staged Configuration Using Feature Models. In *Software Product Lines*, Robert L. Nord (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 266–283.

[53] Conrado Daws. 2005. Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In *1st Int. Colloquium on Theoretical Aspects of Computing (ICTAC) (LNCS)*, Vol. 3407. Springer, 280–294.

[54] Luca de Alfaro. 1997. *Formal Verification of Probabilistic Systems.* Ph.D. Dissertation. Stanford University, Department of Computer Science.

[55] Luca de Alfaro. 1999. Computing Minimum and Maximum Reachability Times in Probabilistic Systems. In *10th International Conference on Concurrency Theory (CONCUR) (Lecture Notes in Computer Science)*, Vol. 1664. 66–81.

[56] Luca de Alfaro and Rupak Majumdar. 2004. Quantitative solution of omega-regular games. *J. Comput. Syst. Sci.* 68, 2 (2004), 374–397.

[57] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. 2017. A Storm is Coming: A Modern Probabilistic Model Checker. In *29th Int. Conf. on Computer Aided Verification (CAV) (LNCS)*, Vol. 10427. Springer, 592–600.

[58] Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. 2011. Decision Problems for Interval Markov Chains. In *5th International Conference on Language and Automata Theory and Applications (LATA) (Lecture Notes in Computer Science)*, Vol. 6638. Springer, 274–285.

[59] Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. 2015. Permissive Controller Synthesis for Probabilistic Systems. *Logical Methods in Computer Science* 11, 2 (2015).

[60] Clemens Dubslaff, Christel Baier, and Sascha Klüppelholz. 2015. Probabilistic Model Checking for Feature-Oriented Systems. *Trans. Aspect-Oriented Software Development* 12 (2015), 180–220.

[61] Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. 2014. Probabilistic Model Checking for Energy Analysis in Software Product Lines. In *13th International Conference on Modularity (MODULARITY)*. ACM, 169–180.

[62] Rüdiger Ehlers and Ufuk Topcu. 2014. Resilience to intermittent assumption violations in reactive synthesis. In *17th International Conference on Hybrid Systems: Computation and Control (HSCC)*. ACM, 203–212.

[63] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. 2008. Multi-Objective Model Checking of Markov Decision Processes. *Logical Methods in Computer Science* 4, 4 (2008).

[64] Kousha Etessami and Mihalis Yannakakis. 2015. Recursive Markov Decision Processes and Recursive Stochastic Games. *J. ACM* 62, 2 (2015), 11.

[65] Jerzy Filar and Koos Vrieze. 1996. *Competitive Markov Decision Processes.*

[66] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. 2016. Supporting Self-Adaptation via Quantitative Verification and Sensitivity Analysis at Run Time. *IEEE Transactions on Software Engineering* 42, 1 (2016), 75–99.

[67] Vojtěch Forejt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. Automated Verification Techniques for Probabilistic Systems. In *11th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM) (Lecture Notes in Computer Science)*, Vol. 6659. 53–113.

[68] Carlo Ghezzi and Amir Molzam Sharifloo. 2013. Model-based verification of quantitative non-functional properties for software product lines. *Information & Software Technology* 55, 3 (2013), 508–524.

[69] Sergio Giro and Pedro R. D'Argenio. 2007. Quantitative Model Checking Revisited: Neither Decidable Nor Approximable. In *5th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS) (Lecture Notes in Computer Science)*, Vol. 4763. Springer, 179–194.

[70] Sergio Giro, Pedro R. D'Argenio, and Luis María Ferrer Fioriti. 2009. Partial Order Reduction for Probabilistic Systems: A Revision for Distributed Schedulers. In *20th International Conference on Concurrency Theory (CONCUR) (Lecture Notes in Computer Science)*, Vol. 5710. Springer, 338–353.

[71] Hassan Gomaa and Mohamed Hussein. 2003. Dynamic Software Reconfiguration in Software Product Families. In *PFE*. 435–444.

[72] Christoph Haase and Stefan Kiefer. 2015. The Odds of Staying on Budget. In *42nd International Colloquium on Automata, Language and Programming (ICALP) (Lecture Notes in Computer Science)*, Vol. 9135. 234–246.

[73] Ernst Moritz Hahn, Tingting Han, and Lijun Zhang. 2011. Synthesis for PCTL in Parametric Markov Decision Processes. In *NASA Formal Methods - Third International Symposium (NFM) (Lecture Notes in Computer Science)*, Vol. 6617. Springer, 146–161.

[74] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. 2010. PARAM: A Model Checker for Parametric Markov Models. In *22nd Int. Conference on Computer Aided Verification (CAV) (LNCS)*, Vol. 6174. Springer, 660–664.

[75] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. 2011. Probabilistic reachability for parametric Markov models. *Int. Journal on Software Tools for Technology Transfer* 13, 1 (2011), 3–19.

[76] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. 2013. eBond: Energy Saving in Heterogeneous R.A.I.N. In *Proc. of the 4th International Conference on Future Energy Systems*. ACM, 193–202.

[77] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. 2008. Approximate Parameter Synthesis for Probabilistic Time-Bounded Reachability. In *29th IEEE Real-Time Systems Symposium (RTSS)*. IEEE Computer Society, 173–182.

[78] Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6 (1994), 512–535.

[79] Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. 2018. Multi-cost Bounded Reachability in MDP. In *24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (Lecture Notes in Computer Science)*, Vol. 10806. Springer, 320–339.

[80] Holger Hermanns, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Markus Siegle. 2003. On the use of MTBDDs for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming* 56, 1-2 (2003), 23–67.

[81] Chung-Hao Huang, Doron A. Peled, Sven Schewe, and Farn Wang. 2016. A Game-Theoretic Foundation for the Maximum Software Resilience against Dense Errors. *IEEE Transactions on Software Engineering* 42, 7 (2016), 605–622.

[82] Lisa Hutschenreiter, Christel Baier, and Joachim Klein. 2017. Parametric Markov Chains: PCTL Complexity and Fraction-free Gaussian Elimination. In *8th Int. Symp. on Games, Automata, Logics and Formal Verification (GandALF) (EPTCS)*, Vol. 256. 16–30.

[83] Nils Jansen, Florian Corzilius, Matthias Volk, Ralf Wimmer, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. 2014. Accelerating Parametric Probabilistic Verification. In *11th Conference on Quantitative Evaluation of Systems (QEST) (LNCS)*, Vol. 8657. Springer, 404–420.

[84] Bengt Jonsson and Kim Guldstrand Larsen. 1991. Specification and Refinement of Probabilistic Processes. In *Sixth Annual Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 266–277.

[85] Lodewijk C.M. Kallenberg. 1983. Linear Programming and Finite Markovian Control Problems. *Mathematical Center Tracts* 148 (1983).

[86] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report. Carnegie-Mellon University Software Engineering Institute.

[87] Shmuel Katz. 1993. A superimposition control construct for distributed systems. *ACM Trans. Program. Lang. Syst.* 15, 2 (April 1993), 337–356.

[88] Joachim Klein, Christel Baier, Philipp Chrszon, Marcus Daum, Clemens Dubslaff, Sascha Klüppelholz, Steffen Märcker, and David Müller. 2018. Advances in probabilistic model checking with PRISM: variable reordering, quantiles and weak deterministic Büchi automata. *International Journal on Software Tools for Technology Transfer* 20, 2 (2018), 179–194.

[89] Jan Kretínský and Tobias Meggendorfer. 2018. Conditional Value-at-Risk for Reachability and Mean Payoff in Markov Decision Processes. In *33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM, 609–618.

[90] Jan Kretínský, Guillermo A. Pérez, and Jean-François Raskin. 2018. Learning-Based Mean-Payoff Optimization in an Unknown MDP under Omega-Regular Constraints. In *29th International Conference on Concurrency Theory (CONCUR) (LIPIcs)*, Vol. 118. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 8:1–8:18.

[91] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *23rd International Conference on Computer Aided Verification (CAV) (Lecture Notes in Computer Science)*, Vol. 6806. 585–591.

[92] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. 2013. Compositional probabilistic verification through multi-objective model checking. *Information and Computation* 232 (2013), 38–65.

[93] André Lanna, Thiago Castro, Vander Alves, Genaina Rodrigues, Pierre-Yves Schobbens, and Sven Apel. 2018. Feature-family-based reliability analysis of software product lines. *Information and Software Technology* 94 (2018), 59 – 81.

[94] Ruggero Lanotte, Andrea Maggiolo-Schettini, and Angelo Troina. 2007. Parametric probabilistic transition systems for system design and analysis. *Formal Aspects of Computing* 19, 1 (2007), 93–109.

[95] Linda Leuschner, Martin Küttler, Tobias Stumpf, Christel Baier, Hermann Härtig, and Sascha Klüppelholz. 2017. Towards Automated Configuration of Systems with Non-Functional Constraints. In *16th Workshop on Hot Topics in Operating Systems (HotOS)*. ACM, 111–117.

[96] William S. Lovejoy. 1991. A survey of algorithmic methods for partially observable Markov decision processes. *Annals of Operations Research* 28, 1 (1991), 47–65.

[97] Ornid Madani, Steve Hanks, and Anne Condon. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence* 147, 1-2 (2003), 5–34.

[98] Rupak Majumdar, Elaine Render, and Paulo Tabuada. 2013. A theory of robust omega-regular software synthesis. *ACM Trans. Embedded Comput. Syst.* 13, 3 (2013), 48:1–48:27.

[99] Steffen Märcker, Christel Baier, Joachim Klein, and Sascha Klüppelholz. 2017. Computing Conditional Probabilities: Implementation and Evaluation. In *15th International Conference on Software Engineering and Formal Methods (SEFM) (Lecture Notes in Computer Science)*, Vol. 10469. Springer, 349–366.

[100] Richard Mayr, Sven Schewe, Patrick Totzke, and Dominik Wojtczak. 2017. MDPs with Energy-Parity Objectives. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (IEEE Computer Society)*. 1–12.

[101] Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. 2000. Complexity of finite-horizon Markov decision process problems. *J. ACM* 47, 4 (2000), 681–720.

[102] Daniel Neider, Alexander Weinert, and Martin Zimmermann. 2018. Synthesizing Optimally Resilient Controllers. In *27th EACSL Annual Conference on Computer Science Logic (CSL) (LIPIcs)*, Vol. 119. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 34:1–34:17.

[103] Christos Papadimitriou and John Tsitsiklis. 1987. The Complexity of Markov Decision Processes. *Mathematics of Operations Research* 12, 3 (1987), 441–450.

[104] Shashank Pathak, Erika Ábrahám, Nils Jansen, Armando Tacchella, and Joost-Pieter Katoen. 2015. A Greedy Approach for the Efficient Repair of Stochastic Models. In *7th International Symposium on NASA Formal Methods (NFM) (Lecture Notes in Computer Science)*, Vol. 9058. Springer, 295–309.

[105] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[106] Mickael Randour, Jean-François Raskin, and Ocan Sankur. 2015. Percentile Queries in Multi-Dimensional Markov Decision Processes. In *27th International Conference on Computer Aided Verification (CAV) (Lecture Notes in Computer Science)*, Vol. 9206. 123–139.

[107] Genaína Nunes Rodrigues, Vander Alves, Vinicius Nunes, André Lanna, Maxime Cordy, Pierre-Yves Schobbens, Amir Molzam Sharifloo, and Axel Legay. 2015. Modeling and Verification for Probabilistic Properties in Software Product Lines. In *16th IEEE International Symposium on High Assurance Systems Engineering (HASE)*. IEEE Computer Society, 173–180.

[108] Ina Schaefer, Lorenzo Bettini, Viviana Bono, Ferruccio Damiani, and Nico Tanzarella. 2010. Delta-Oriented Programming of Software Product Lines. In *Software Product Lines: Going Beyond*, Jan Bosch and Jaejoon Lee (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 77–91.

[109] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2006. Model-Checking Markov Chains in the Presence of Uncertainties. In *12th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (LNCS)*, Vol. 3920. Springer, 394–410.

[110] James P.G. Sterbenz, David Hutchison, Egemen K. Çetinkaya, Abdul Jabbar, Justin P. Rohrer, Marcus Schöller, and Paul Smith. 2010. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks* 54, 8 (2010), 1245 – 1265.

[111] Guoxin Su, Taolue Chen, Yuan Feng, David S. Rosenblum, and P. S. Thiagarajan. 2016. An Iterative Decision-Making Scheme for Markov Decision Processes and Its Application to Self-adaptive Systems. In *19th International Conference on Fundamental Approaches to Software Engineering (FASE) (Lecture Notes in Computer Science)*, Vol. 9633. Springer, 269–286.

[112] Guoxin Su, David S. Rosenblum, and Giordano Tamburrelli. 2016. Reliability of Run-Time Quality-of-Service evaluation using parametric model checking. In *38th International Conference on Software Engineering (ICSE)*. ACM, 73–84.

[113] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. 2014. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Comput. Surv.* 47, 1s (2014), 6:1–6:45.

[114] Michael Ummels and Christel Baier. 2013. Computing Quantiles in Markov Reward Models. In *16th International Conference on Foundations of Software Science and Computation Structures (FOSSACS) (Lecture Notes in Computer Science)*, Vol. 7794. 353–368.

[115] Moshe Y. Vardi. 1985. Automatic verification of probabilistic concurrent finite-state programs. In *26th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 327–338.

[116] Moshe Y. Vardi and Pierre Wolper. 1986. An automata-theoretic approach to automatic program verification (preliminary report). In *1st Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 332–344.

[117] Christian von Essen, Barbara Jobstmann, David Parker, and Rahul Varshneya. 2016. Synthesizing efficient systems in probabilistic environments. *Acta Inf.* 53, 4 (2016), 425–457.