Constrained and Robust Policy Synthesis with Satisfiability-Modulo-Probabilistic-Model-Checking

Linus Heck[1], **Filip Macák**[2], **Milan Češka**[2], **Sebastian Junges**[1]
[1]Radboud University, Nijmegen, The Netherlands
[2]Brno University of Technology, Brno, Czech Republic
{linus.heck, sebastian.junges}@ru.nl, {imacak, ceskam}@fit.vut.cz

Abstract

The ability to compute reward-optimal policies for given and known finite Markov decision processes (MDPs) underpins a variety of applications across planning, controller synthesis, and verification. However, we often want policies (1) to be robust, i.e., they perform well on perturbations of the MDP and (2) to satisfy additional structural constraints regarding, e.g., their representation or implementation cost. Computing such robust and constrained policies is indeed computationally more challenging. This paper contributes the first approach to effectively compute robust policies subject to arbitrary structural constraints using a flexible and efficient framework. We achieve flexibility by allowing to express our constraints in a first-order theory over a set of MDPs, while the root for our efficiency lies in the tight integration of satisfiability solvers to handle the combinatorial nature of the problem and probabilistic model checking algorithms to handle the analysis of MDPs. Experiments on a few hundred benchmarks demonstrate the feasibility for constrained and robust policy synthesis and the competitiveness with state-of-the-art methods for various fragments of the problem.

1 Introduction

Markov decision processes (MDPs) are a prominent model for sequential decision making under uncertainty. For a given MDP, a standard planning task is to compute a policy that optimizes some value, e.g., a discounted reward over an infinite horizon (Puterman 1994). Optimal policies for such tasks can be efficiently computed both in theory and in practice. However, not every policy that can be computed can be deployed: Policies are often subject to additional constraints, in particular (1) *structural constraints* that require a certain shape or structure of the *controller* representing the policy and policies should be (2) *robust* against perturbations of the environments. Structural constraints may include constraints that require that a policy can be encoded as a small decision tree, see e.g., (Ashok et al. 2021; Vos and Verwer 2023) and constraints that enforce that the policy is observation-based, as in partially observable MDPs (POMDPs) (Smallwood and Sondik 1973). Perturbations can be used to express that a policy achieves sufficient value in a set of MDPs, e.g, (Chades et al. 2012; Ghezzi and Sharifloo 2013). The key problem statement we work towards is:

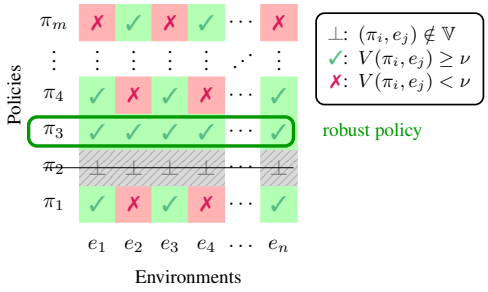


Figure 1: One axis represents the set of policies π_i, the other the set of possible environments e_j. Some policies (or environments) may not be relevant ($\perp$), e.g. when the policy does not satisfy structural constraints. For all tuples of relevant policy π_i and environment e_j, the combination performs satisfactory (✓) or unsatisfactory (✗). Satisfactory performance is captured by meeting a threshold ν on the value of the policy on the environment. The general task is to find a policy π_i such that for every environment e_j, cell (i, j) is ✓. While this figure uses a tabular representation, we use concise symbolic descriptions.

Is there a policy, subject to structural constraints, such that on every (PO)MDP (environment) in a set of (PO)MDPs, its value is sufficient? Fig. 1 illustrates this problem statement.

Accommodating structural constraints and robustness brings two challenges: (1) Already when considering *either* structural constraints *or* robustness, we must deal with a *combinatorial* problem. For various decision problems that underpin planning with additional constraints on the structure of the policy or on the robustness against perturbation, (co)NP-hardness results exist (Andriushchenko et al. 2025a; van der Vegt, Jansen, and Junges 2023; Chatterjee, Doyen, and Henzinger 2010). Typically, reasoning about every policy satisfying the constraints or every possible perturbation is necessary *in the worst case*. (2) To support robustness, we must implicitly or explicitly deal with a *quantifier alternation*. The key problem statement existentially quantifies the policy and universally quantifies over the environments.

A classical approach to deal with the combinatorial nature of the problems is to create a mixed integer linear program (MILP), where the integer variables encode the choices for

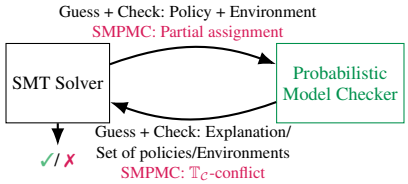
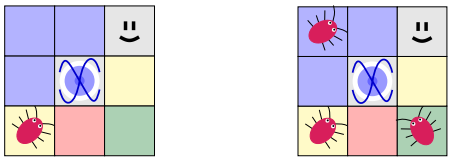
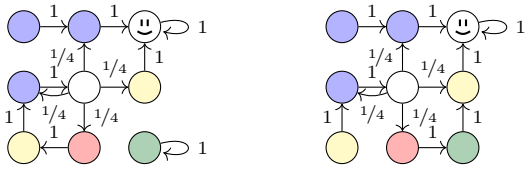


Figure 2: A separation of concerns using guess-and-check or counterexample-guided-inductive-synthesis.



(a) Single initial location (b) Multiple initial locations

Figure 3: Running example: Beetle wants to reach the target.



(a) $\{d_r{=}1, d_g{=}3, d_b{=}3, d_y{=}0\}$ solves single initial location (b) $\{d_r{=}3, d_g{=}0, d_b{=}3, d_y{=}0\}$ solves multiple initial locations

Figure 4: Markov chains induced by different beetle policies.

the policies and the perturbations, while an LP encodes the value of the resulting MDP (Kumar and Zilberstein 2015a; Vos and Verwer 2023). Such monolithic encodings have various drawbacks, in particular, using LP solvers is typically much slower than value- or policy-iteration based approaches to solving MDPs, see e.g. (Hartmanns et al. 2023).

The state-of-the-art, therefore, avoids monolithic encodings by applying a separation of concerns (Andriushchenko et al. 2025b). It instantiates a typical guess-and-check (or inductive synthesis) approach, which we briefly summarize here. Consider the black loop in Fig. 2: A satisfiability (SAT) solver picks a policy satisfying the structural constraints (or analogously, the environment) and then the policy (and environment) is evaluated by a dedicated numerical routine, in this case, the probabilistic model checking (PMC) framework Storm (Hensel et al. 2022). Without further adaptation, such an approach would need to enumerate all policies. However, Storm can generalize the results from an individual policy to a set of policies (Češka et al. 2021). The separation allows using modern SAT solvers for the combinatorial problem and highly efficient routines for policy evaluation. Still, this existing approach has at least two shortcomings. First, generalizing from one policy to a set of policies is often expensive, yet necessary to avoid enumerating the policy space. Second, support for quantifier alternations is not straightforward as the PMC framework would need to evaluate any policy on *every* environment explicitly. For some classes of simple constraints, dedicated branch-and-bound style approaches have been developed with superior performance, e.g., to compute fixed-memory policies for POMDPs (Andriushchenko et al. 2023).

This paper presents SAT-Modulo-PMC (SMPMC, say *sampick*), a novel synthesis approach that (1) avoids the expensive generalization step crucial for the CEGIS approach, while (2) providing principled support for simultaneous structural constraints and robustness. The approach is (3) sufficiently flexible to support, for the first time, a complete approach that can find provably smallest decision tree encodings of robust policies. Technically, the key innovation in SMPMC is a tight integration of modern satisfiability solvers and PMC. More precisely, we consider Z3 (de Moura and Bjørner 2008), a satisfiability modulo theories solver with support for quantified logics and the recently developed option to add custom theories (Bjørner, Eisenhofer, and Kovács 2023). We develop a custom theory based on probabilistic model checking. Algorithmically, SMPMC can be seen as a powerful branch-and-bound variation where the SAT solver realizes intelligent branching and the probabilistic model checker ensures effective bounding. Despite its flexibility, the novel technique is highly competitive and it outperforms a monolithic approach to robust and constrained synthesis by orders of magnitude.

Contributions. In summary, we present SMPMC, a novel, versatile, and efficient approach that tightly integrates satisfiability solvers with probabilistic model checking. SMPMC is the first effective approach that constructs decision tree representations of robust policies and is even able to prove the absence of a fixed-size robust decision tree. For various fragments of the robust constrained synthesis problem, SMPMC is generally at least competitive with ad-hoc state-of-the-art heuristics and solves many benchmarks that the state-of-the-art cannot solve.

2 Setting and Motivating Example

In this section, we introduce a motivating example that we refer to throughout this paper. We first introduce the problem and then work towards a formalization.

Problem description. Consider Fig. 3: The agent (depicted as a beetle) moves in a tiny grid-world environment including the target in the north east corner, the only cell equipped with a reward, and a fountain in the middle, which will spray the beetle in a random direction. We impose the following simplified constraints: (1) the beetle must move in the same direction in cells with the same floor color, and (2) must go in a different direction on blue tiles than on yellow tiles. Our goal is to find a memoryless policy that eventually collects the reward. Fig. 3a illustrates this problem for a known initial location of the agent. This problem corresponds to *structurally constrained* synthesis of (memoryless) policies in (PO)MDPs. If the agent must use a single policy that is robust against perturbations of the initial state (see Fig. 3b), the problem becomes harder and corresponds to a constrained *and robust* policy synthesis in (PO)MDPs.

Challenges. Two aspects make the example challenging:

1. Some policy choices (e.g., the beetle goes right on blue tiles) can affect multiple state-action pairs in the model (e.g., the direction for all blue tiles).
2. Some constraints on policy or environment choices (e.g., the beetle goes in different directions on blue and yellow tiles) involve multiple choices at once.

More generally, the first aspect is crucial when reasoning about robust policies for multiple environments, where the agent cannot observe which environment is describing the world. In those cases, a robust policy must take the same actions in all environments. This adds structurally simple identities between action choices in different states, similar to observation-based memoryless policies in POMDPs. Beyond these structurally simple identities, the second property is a simple type of additional constraint.

Problem modeling. We reason about the policy and environment in a unified parameter space, where policies are given using *controllable parameters* $X = \{d_r, d_g, d_b, d_y\}$ (i.e., the direction for each color). We can impose constraints such as $d_b \neq d_y$. To represent the environments, we use *uncontrollable parameters* $Y = \{s_x, s_y\}$ (i.e., representing the starting coordinates). The beetle's policy must be robust against the choice of the parameters in Y. The possible starting locations are restricted by the constraint $(s_x = 0 \wedge s_y = 0) \vee (s_x = 2 \wedge s_y = 0) \vee (s_x = 0 \wedge s_y = 2)$. In general, constraints can be represented by arbitrary logical formulas over $X \cup Y$. A satisfying MC (corresponding to a satisfying assignment) for the single initial location can be seen in Fig. 4a. In the case of multiple initial locations, we have a quantifier alternation: we ask whether there exists an assignment of the controllable parameters that is satisfying for all assignments of the uncontrollable parameters. A satisfying MC (where a reward 1 is collected from each initial state) is shown in Fig. 4b.

Notation. We present our approach using generic (un)controllable parameters X, Y and assignments rather than using policies and environments.

3 Problem Statement

We formalize the main problem statement. We first recap Markov decision processes (MDPs) and colored MDPs.

Definition 1 (Markov Decision Process)**.** *A* Markov decision process (MDP) *is a tuple $\mathcal{M} = (S, s_I, Act, \mathcal{P}, R)$ with a finite set S of states, an initial state $s_I \in S$, a finite set Act of actions, a partial transition function $\mathcal{P}\colon S \times Act \rightharpoonup Distr(S)$ and a state reward function $R : S \to \mathbb{R}_{\geq 0}$.*

We write $Act(s) := \{\alpha \in Act \mid \mathcal{P}(s, \alpha) \neq \bot\}$ for *available actions* at s. An MDP policy is a function $\pi\colon S \to Act$ s.t. $\pi(s) \in Act(s)$ for all $s \in S$. A Markov chain (MC) is an MDP where $|Act(s)| = 1$ for all $s \in S$. We denote their transition functions as $\mathcal{P}\colon S \to Distr(S)$. For MDP $\mathcal{M}$ and policy π, we define the induced MC $\mathcal{M}[\pi] = (S, s_I, \mathcal{P}', R)$ with $\mathcal{P}'(s) = \mathcal{P}(s, \pi(s))$ for any $s \in S$.

Specification. We consider undiscounted indefinite-horizon expected rewards (Puterman 1994). These generalize both finite-horizon rewards and infinite-horizon discounted rewards. Given an MC $\mathcal{D}$, a *path* ξ in $\mathcal{D}$ is an infinite sequence of states $s_0 s_1 s_2 \ldots \in S$ where $s_0 = s_I$. We define $R(\xi) = \sum_{i=0}^{\infty} R(s_i)$ as the (possibly infinite) reward of a path. We define the *value* of D as $V(\mathcal{D}) := \mathbb{E}[R(\xi)]$ if the expected value exists and ∞ otherwise. For an MDP $\mathcal{M}$, we define the *maximal value* $V^{\max}(\mathcal{M}) := \max_\pi V(\mathcal{M}[\pi])$ and the *minimal value* $V^{\min}$ analogously.

Parameter space. We introduce a *parameter space* to symbolically reason about policies and environments. Let X be a set of *controllable parameters* and Y be a set of *uncontrollable parameters*, representing, e.g., policies and environments, respectively. The sets X and Y are disjoint. We denote assignments to variables $X \cup Y$ by values from $\mathbb{Z}$ with $\mathbb{Z}^{X \cup Y}$. We consider a finite *constrained parameter space* $\mathbb{V} \subseteq \mathbb{Z}^{X \cup Y}$. In the remainder, θ denotes assignments in $\mathbb{V}$, while $\eta \subseteq \mathbb{V}$ denotes a nonempty subset of $\mathbb{V}$ and is referred to as a *set of assignments*. We write $\eta(p) = \{\theta(p) \mid \theta \in \eta\}$ for parameter $p \in X \cup Y$. In this paper, we represent $\mathbb{V}$ as the satisfying assignments of a logical formula.

Projected assignments. For $Z \in \{X, Y\}$ and $\theta \in \mathbb{V}$, the projected assignment $\theta_Z \in \mathbb{Z}^Z$ coincides with θ on the variables Z. We use such projections to write assignment θ as $\theta_X \theta_Y$ and lift them to the parameter space: $\mathbb{V}_X := \{\theta_X \in \mathbb{Z}^X \mid \exists \theta_Y : \theta_X \theta_Y \in \mathbb{V}\}$. $\mathbb{V}_Y$ is defined analogously.

Colored MDPs. We represent finite sets of MCs by a *colored MDP* (Andriushchenko et al. 2025b). A colored MDP $\mathcal{C}$ relates to the constrained parameter space $\mathbb{V}$ by mapping state-action pairs (s, a) in $\mathcal{C}$ to a set of assignments $\eta \subseteq \mathbb{V}$. This represents for which $\theta \in \eta$ distribution (s, a) exists:

Definition 2 (Colored MDP)**.** *A* colored MDP $\mathcal{C} = (\mathcal{M}, \mathbb{V}, \kappa)$ *consists of an MDP $\mathcal{M}$ with states S and actions Act, a constrained parameter space $\mathbb{V}$, and a* coloring $\kappa(s, \alpha) \subseteq \mathbb{V}$*, such that for all $\theta \in \mathbb{V}, s \in S$, there is exactly one action $\alpha \in Act(s)$ with $\theta \in \kappa(s, \alpha)$.*

Running Example 1. *For Fig. 3a, suppose $s \in S$ is a blue state with $Act(s) = \{0, 1, 2, 3\}$ representing the directions. Then $\kappa(s, \alpha) = \{\theta \mid \theta(d_b) = \alpha\}$ for $\alpha \in Act(s)$. In Fig. 4a, we have $\theta(d_b) = 3$, which selects action 3 in all blue states.*

Given a colored MDP $\mathcal{C}$ on $\mathbb{V}$ and a set of assignments η, the *induced MDP* $\mathcal{C}[\eta]$ restricts $\mathcal{C}$ to actions admissible by η:

Definition 3 (Induced MDP)**.** *For a set of assignments $\eta \subseteq \mathbb{V}$, the* induced MDP $\mathcal{C}[\eta]$ *is the largest sub-MDP $\mathcal{M}'$ of $\mathcal{M}$ with actions Act' s.t. $\eta \cap \kappa(s, \alpha) \neq \emptyset$ for all $s \in S, \alpha \in Act'(s)$. For $\theta \in \mathbb{V}$, the MDP $\mathcal{C}[\theta] := \mathcal{C}[\{\theta\}]$ is an MC.*

We can now formally state the problem this paper tackles:

> **Problem Statement** (Robust Constrained Feasibility). Given a colored MDP $\mathcal{C}$ with constrained parameter space $\mathbb{V}$ on controllable parameters X and uncontrollable parameters Y, and a threshold $\nu \in \mathbb{R}$, does there exist an assignment $\theta_X \in \mathbb{V}_X$ s.t for all assignments $\theta_Y \in \mathbb{V}_Y$ with $\theta_X \theta_Y \in \mathbb{V}$: $V(\mathcal{C}[\theta_X \theta_Y]) \geq \nu$?

We thus ask for a policy, encoded as θ_X, that satisfies the value threshold ν in all environments, encoded by θ_Y. In the remainder, we assume a fixed ν and omit it for conciseness.

4 The SMPMC Approach

We introduce *Satisfiability Modulo Probabilistic Model Checking* (SMPMC, pronounced *sampick*). We first define a first-order theory that encodes that the value of a colored MDP at an assignment is satisfactory and show that the validity of specific sentences in this theory corresponds to solving the problem statement. We then discuss how our novel SMPMC approach integrates this theory into the lazy constraint-driven conflict learning (CDCL) method. Finally, we describe how to extend this method to solve robust constrained synthesis.[1]

4.1 First-Order Formulation

Given an instance of the problem statement on a colored MDP $\mathcal{C}$ and let $X \cup Y = \{p_1, p_2, \ldots, p_n\}$. We formulate the problem statement in a dedicated first-order theory $\mathbb{T}_\mathcal{C}$ over bounded integer variables $X \cup Y$, which introduces a single n-ary predicate viable over $X \cup Y$. Intuitively, the atom $\mathsf{viable}(\theta(p_1), \ldots, \theta(p_n))$ evaluates to true iff $V(\mathcal{C}[\theta]) \geq \nu$. Formally, we uniquely characterize the theory $\mathbb{T}_\mathcal{C}$ by the set of true sentences. The true sentences are the deductive closure of the following sets of sentences:

$$\{\mathsf{viable}(\theta(p_1), \ldots, \theta(p_n)) \quad \mid \theta \in \mathbb{V} \text{ with } V(\mathcal{C}[\theta]) \geq \nu\},$$
$$\{\neg\mathsf{viable}(\theta(p_1), \ldots, \theta(p_n)) \quad \mid \theta \in \mathbb{V} \text{ with } V(\mathcal{C}[\theta]) < \nu\}.$$

In particular, this ensures that for an assignment $\theta \in \mathbb{V}$, the first-order sentence $\mathsf{viable}(\theta(p_1), \ldots, \theta(p_n))$ is true in $\mathbb{T}_\mathcal{C}$ iff the colored MDP $\mathcal{C}$ has a value of at least ν at θ.

By including the theory of the bounded integers (Biere et al. 2009, p831), we can use a formula $\tau_\mathbb{V}$ to represent the constraints of the constrained parameter space $\mathbb{V}$ logically. Given the logical formula $\tau_\mathbb{V}$, the problem statement can be rephrased as a first-order sentence:

$$\Phi := \exists x_1 \ldots \exists x_n \; \forall y_1 \ldots \forall y_m : \tau_\mathbb{V}(x_1, \ldots, x_n, y_1, \ldots, y_m) \\ \rightarrow \mathsf{viable}(x_1, \ldots, x_n, y_1, \ldots, y_m). \quad (1)$$

Theorem 4. *For a colored MDP $\mathcal{C}$, the assignment $\theta_X \in \mathbb{V}_X$ over variables $X = \{x_1, \ldots, x_n\}$ is a satisfying assignment of the existential quantifier in the sentence Φ if and only if θ_X is a solution to the problem statement.*

4.2 CDCL and Theory Solvers

We first describe SMPMC assuming that $Y = \emptyset$ and leave the evaluation of the universal quantifier for Section 4.4. The core algorithm in most SMT solvers is *Conflict-Driven Clause Learning (CDCL)* (Bayardo Jr. and Schrag 1997; Silva and Sakallah 1999). The goal of CDCL is to find a satisfying assignment of the existentially quantified variables in our input formula Φ. To do this, CDCL iteratively constructs a partial assignment K of these variables[2]. At each step of this iteration, CDCL invokes a subroutine to check whether any full extension of K could satisfy Φ. If this is not the case, we call K a *conflict*. In CDCL with theories

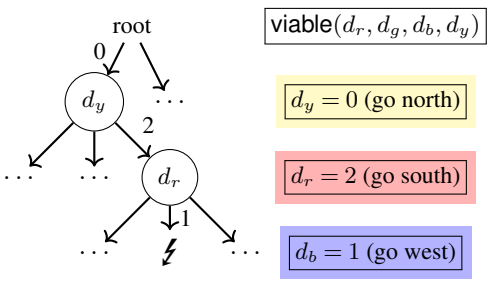


Figure 5: Sketch of CDCL.

(Nieuwenhuis, Oliveras, and Tinelli 2006), all theories involved in the assignment K provide a *theory solver* that is queried and asked whether K is a conflict within that theory. In SMPMC, we provide a theory solver for the theory $\mathbb{T}_\mathcal{C}$ introduced above. Fig. 2 describes the loop in red: (1) The SMT solver provides the theory solver with K. (2) The theory solver reports whether K is a conflict.

We match a partial assignment K to a set of first-order *literals* that assign variables to values. In the following, we use both interchangeably. A partial assignment K is *consistent* if there exists a full variable assignment θ that makes all literals in K true, i.e., θ is a full extension of the partial assignment K. The set K is *inconsistent* if there is no such variable assignment. We call a partial assignment K a $\mathbb{T}_\mathcal{C}$*-conflict* if $K \cup \mathbb{T}_\mathcal{C}$ is inconsistent. If the theory solver proves K to be a $\mathbb{T}_\mathcal{C}$-conflict, it propagates a conflict back to the SMT solver. It repeats this process until it finds a satisfying assignment or proves Φ false. We illustrate the CDCL algorithm combined with our theory solver in the following example.

Running Example 2. *Fig. 5 shows a hypothetical run of CDCL where it has fixed three variables. The partial assignment is: $K = \{\mathsf{viable}(d_r, d_g, d_b, d_y), d_y{=}0, d_r{=}2, d_b{=}1\}$. This means that in Fig. 3a, the beetle must go north on yellow tiles, south on red tiles, and west on blue tiles. The theory solver proves that K is a $\mathbb{T}_\mathcal{C}$-conflict.*

From now on, K contains only $(\neg)\mathsf{viable}(p_1, \ldots, p_n)$ and assignments of the variables $p_1, \ldots, p_n \in X \cup Y$. The assignment $\theta \in \mathbb{V}$ satisfies K if it satisfies its conjunction.

4.3 Theory Solver for $\mathbb{T}_\mathcal{C}$

The theory solver wants to prove that a partial assignment K is a $\mathbb{T}_\mathcal{C}$-conflict. As $\mathbb{V}$ is finite, one correct but intractable approach to decide whether a given $K \cup \mathbb{T}_\mathcal{C}$ is (in)consistent is to enumerate all possible assignments. Instead, the theory solver outlined in this section checks an induced MDP as an *abstraction* of the colored MDP $\mathcal{C}$. If the value of the induced MDP is not within the threshold ν, it derives a $\mathbb{T}_\mathcal{C}$-conflict:

Theorem 5. *Let $\mathcal{C}$ be a colored MDP and K be a partial assignment to Φ s.t. $v := \mathsf{viable}(p_1, \ldots, p_n) \in K$. Let $\eta = \{\theta \mid \theta \text{ satisfies } K \backslash \{v\}\}$. If $V^{\max}(\mathcal{C}[\eta]) < \nu$, then K is a $\mathbb{T}_\mathcal{C}$-conflict. The same is true for $\neg v \in K$ and $V^{\min}(\mathcal{C}[\eta]) \geq \nu$.*

Theorem 5 follows from (Andriushchenko et al. 2025b). The intuition is that if $V^{\max}(\mathcal{C}[\eta]) < \nu$, then all induced MCs of the MDP $\mathcal{C}[\eta]$ have a value below ν. Thus, there is no

[1] We provide an extended example in Appendix B.1 and prove all theorems in Appendix B.2 (https://arxiv.org/abs/2511.08078).

[2] Strictly speaking, our variables are bit vectors. We then say a variable is assigned when all bits are assigned.

assignment $\theta \in \eta$ such that $V(\mathcal{C}[\theta]) \geq \nu$. Note that because deciding whether K is a $\mathbb{T}_\mathcal{C}$-conflict is NP-hard, Theorem 5 only gives us a complete method if η contains a single θ.

SMT solvers profit from theory solvers that find small conflicts (Biere et al. 2009, p849). In particular, a $\mathbb{T}_\mathcal{C}$-conflict K may contain statements that do not contribute to being one. Suppose that we know that K is a $\mathbb{T}_\mathcal{C}$-conflict, we now aim to find a smaller $\mathbb{T}_\mathcal{C}$-conflict $C(K) \subseteq K$.

We say that a coloring $\kappa(s, \alpha)$ is *independent* of a parameter p_i if the value of that parameter is irrelevant for membership in $\kappa(s, \alpha)$. If *all* colorings in $\mathcal{C}$ that are reachable in $\mathcal{C}[\eta]$ are independent of p_i, it cannot contribute to K being a $\mathbb{T}_\mathcal{C}$-conflict. We can thus remove all assignments of independent parameters from K, yielding a *restricted partial assignment* $C(K)$ over the remaining assignments. We define independence and $C(K)$ precisely in Appendix B.2.

Theorem 6. *Given $\mathcal{C}$ and K from Theorem 5, the restricted partial assignment $C(K) \subseteq K$ is also a $\mathbb{T}_\mathcal{C}$-conflict.*

Theorem 6 describes a computationally efficient way to compute a smaller $\mathbb{T}_\mathcal{C}$-conflict by computing the independent parameters. In contrast, the *counterexample generalization* on full assignments in (Češka et al. 2021) is NP-hard.

Running Example 3. *In Fig. 5, the theory solver checks whether for $\eta = \{\theta \mid \theta(d_y) = 0, \theta(d_r) = 2, \theta(d_b) = 1\}$, we have $V^{\max}(\mathcal{C}[\eta]) < 1$. This is true, so K is a $\mathbb{T}_\mathcal{C}$-conflict because of Theorem 5. As all colorings reachable in $\mathcal{C}[\eta]$ are independent of d_r, the theory solver can compute a smaller $\mathbb{T}_\mathcal{C}$-conflict $C(K) = \{\mathsf{viable}(d_r, d_g, d_b, d_y), d_y{=}0, d_b{=}1\}$.*

4.4 Handling Quantifier Alternation

We now consider that $Y \neq \emptyset$. Given the quantified statement Φ from Eq. (1), the SMT solver uses *model-based quantifier instantiation (MBQI)* to instantiate the universally quantified variables until it either finds a satisfying assignment for the entire formula or proves it false (Ge and de Moura 2009).

Given Φ from Eq. (1), MBQI first finds a candidate variable assignment θ_X for the existentially quantified variables $x_1, \ldots, x_n$. Then, it substitutes these values into the quantifier-free formula $\phi := \tau_\mathbb{V}(\ldots) \rightarrow \mathsf{viable}(\ldots)$, yielding the formula $\phi^{\theta_X}(y_1, \ldots, y_m)$ over the variables in Y. Next, MBQI checks whether $\neg\phi^{\theta_X}$ is inconsistent. If it is inconsistent, then θ_X is a satisfying assignment of Φ's existential quantifier, as there is no instantiation of $y_1, \ldots, y_n$ that makes ϕ^{θ_X} false. Otherwise, there is a satisfying variable assignment θ_Y of $\neg\phi^{\theta_X}$. MBQI adds a new conflict $\{\phi(x_1^{\theta_X}, \ldots x_n^{\theta_X}, y_1^{\theta_Y}, \ldots, y_n^{\theta_Y})\}$, which rules out at least θ_X (and might rule out more). It repeats this process until either a satisfying assignment is found or Φ is proven false.

Running Example 4. *Given the environment in Fig. 3b, suppose CDCL chooses $\theta_X = \{d_r{=}1, d_g{=}3, d_b{=}3, d_y{=}0\}$. MBQI checks whether $\neg\phi^{\theta_X}(s_x, s_y)$ is inconsistent. As the beetle starting in the bottom right corner will not reach the target in the MC depicted in Fig. 4a, $\{s_x{=}2, s_y{=}0\}$ satisfies $\neg\phi^{\theta_X}(s_x, s_y)$. MBQI pushes the conflict $\{\phi(1, 3, 3, 0, 2, 0)\}$, ruling out θ_X. CDCL chooses $\theta'_X = \{d_r = 3, d_g = 0, d_b = 3, d_y{=}0\}$ (see Fig. 4b). As $\neg\phi^{\theta'_X}(s_x, s_y)$ is inconsistent, the beetle will reach the target no matter where it starts with θ'_X.*

5 Experiments

We investigate the performance of the proposed policy synthesis algorithm SMPMC. The implementation and all the considered benchmarks are available[3].

Research questions. We focus on the following questions:
Q1: *Can SMPMC solve a class of synthesis problems that goes beyond the capabilities of existing tools?*
Q2: *Is SMPMC competitive with state-of-the-art methods for various fragments of the synthesis problem?*

SMPMC implementation. We implemented SMPMC on top of the probabilistic model checker Storm (Hensel et al. 2022) and the SMT solver Z3 (de Moura and Bjørner 2008), leveraging Z3's user propagator API (Bjørner, Eisenhofer, and Kovács 2023). We use PAYNT (Andriushchenko et al. 2025b) to parse the input. Appendix A gives details.

Benchmarks. To systematically answer the research questions, we define four classes **C1–C4** of synthesis problems, see Table 1. They represent different, NP-hard, fragments of the problem statement. The classes **C2–C4** are special cases due to additional assumptions on the constraint $\tau_\mathbb{V}$ and the uncontrollable parameters Y. We consider 372 benchmarks from different domains: (1) sets of MCs, given by parametrized finite-state probabilistic programs, mostly from (Andriushchenko et al. 2025b), (2) synthesis of fixed-memory finite-state controllers (FSCs) for POMDPs from (Andriushchenko et al. 2023), (3) multiple-environment (ME) MDPs from (Andriushchenko et al. 2024; van der Vegt, Jansen, and Junges 2023) that model variations of a system in different environments (Chades et al. 2012), and (4) ME-POMDPs from (Galesloot et al. 2025). Finally, (5) we extended some standard POMDP from (2) to ME-POMDPs. These benchmarks were translated to colored MDPs akin to (Andriushchenko et al. 2025b). Our benchmarks include a similar number of satisfiable and unsatisfiable instances.

Setup. All experiments ran on a single core of an AMD Ryzen TRP 5965WX with a 30-minute timeout and 16GB of available memory. The considered baselines are explained in the *baselines* paragraphs in Q1 and Q2 sections. All tools use optimistic value iteration (Hartmanns and Kaminski 2020) with a precision of 10^{-4} for PMC. We ran each experiment three times and confirmed that Table 2 is the same. All of the considered methods are deterministic and have insignificant variance in timing on fixed hardware.

Main result: Table 2 reports the total number of solved benchmarks across all considered methods and problem classes. In **C1**, only SMPMC solves a significant part of the benchmarks, and for all other classes, it is competitive and solves some previously unsolvable instances.

Below, we detail the results in the context of Q1 and Q2.

Q1: Going Beyond Capabilities of Existing Tools

Setting. Inspired by the recent efforts to find robust policies (Galesloot et al. 2025) and interpretable policies repre-

[3]Code, benchmarks: https://doi.org/10.5281/zenodo.17573007

	S.C.*	$\tau_{\mathbb{V}} \neq \top$	$Y \neq \emptyset$	**Problem Statements: Is there a...**	**Baselines**
C1	✓	✓	✓	decision tree policy that works in all environments?	`SMT(LRA)`
C2	✓	✓	✗	decision tree policy in a POMDP? satisfying, cost-bounded MC in a set of MCs?	`PAYNT-CEGIS`, `SMT(LRA)`
C3	✓	✗	✓	policy in a POMDP that works in all environments? satisfying MC that is robust against perturbations?	`PAYNT-AR` (extended), `SMT(LRA)`
C4	✓	✗	✗	policy in a POMDP? satisfying MC in a set of MCs?	`PAYNT-{AR,CEGIS}`, `SMT(LRA)`

Table 1: Summary of configurations, problem statements, and baselines we compare with. The policies are memoryless or fixed-memory. *All problem categories exhibit structural constraints, e.g., in the form of partial observability.

	#Benchmarks	SMPMC (New!)	`PAYNT-AR`	`PAYNT-CEGIS`	`PAYNT-Hybrid`	`SMT(LRA)`
C1	112	**77 (53)**	N/A	N/A	N/A	24 (0)
C2	72	**54 (1)**	N/A	**54 (1)**	N/A	31 (0)
C3	82	**43 (16)**	**43 (16)**	N/A	N/A	7 (0)
C4	106	**88** (5)	86 (**8**)	24* (0)	23* (0)	30 (0)
$\sum$	372	**262 (75)**	129 (24)	78 (1)	23 (0)	92 (0)

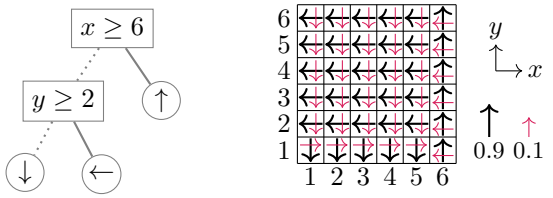
Table 2: The total number of solved benchmarks with the 30-minute timeout for each benchmark. The numbers in parentheses show the number of benchmarks that no other tool has solved. N/A indicates that the method does not support the problem class. *Due to limitations in counterexample generalization, 57 benchmarks are not supported by `PAYNT-CEGIS`, `Paynt-Hybrid`.

sented by small decision trees (Vos and Verwer 2023), we combine both these requirements and seek for robust decision trees (DTs) in various ME-(PO)MDP benchmarks. We formulate an SMT constraint $\tau_{\mathbb{V}}$ that specifies that the policy can be represented by a DT with n nodes (the encoding is described in Appendix D.1). We consider $n \in \{1, 3, \ldots, 15\}$ for all benchmarks.

Baseline. To our best knowledge, there is no tool that is able to solve **C1** problems. Therefore, we implemented a baseline, further denoted as `SMT(LRA)`, that supports these problems by constructing a monolithic SMT query with *linear real arithmetic* and running Z3.

Results. Table 2 shows that SMPMC clearly outperforms the `SMT(LRA)` baseline: It solves all instances that `SMT(LRA)` does and it additionally solves 53 benchmarks that are unfeasible for `SMT(LRA)`. More detailed results presented in Appendix D.1 show that `SMT(LRA)` mostly solves only simple instances of the problems and, on average, SMPMC can solve these instances 20x faster. Moreover, SMPMC is able to solve some very challenging problems, e.g., a benchmark where the robust minimal DT has 15 nodes. In many cases, the underlying colored MDP has above 20K states and the number of unconstrained assignments goes beyond 10^{100} (see Appendix D.3).

Demonstration. We now describe the results for the *Rocks-6-4* ME-MDP taken from (Andriushchenko et al. 2024). The goal of the agent is to collect four rocks on a 6x6 grid, where the environment encodes the position of the rocks. While the authors acknowledged that "there exists a single policy that is winning for all MDPs", their algorithm that searches for unconstrained memoryless policies using a game-based abstraction and various heuristics cannot find



(a) Robust policy (b) Grid, red arrows are slip direction

Figure 6: Minimal decision tree for *Rocks-6-4*, by SMPMC.

it. In contrast, SMPMC finds a minimal 5-node DT that describes a robust policy, shown in Fig. 6a. The policy visits all grid cells, thereby ensuring it collects all rocks. It achieves this in an interesting way: in the *Rocks-6-4* model, the agent slips to the left of the current movement direction with probability 10%. The tree uses this to create a *spiral* effect, visiting all cells eventually, as visualized in Fig. 6b.

Summary Q1. The results obtained for the synthesis problems in **C1** demonstrate that SMPMC solves a class of problems that goes beyond the capabilities of existing tools.

Q2: Comparison on Problem Fragments

Baselines. Besides the monolithic `SMT(LRA)` baseline used in Q1, more baselines are applicable to Q2. We compare against the state-of-the-art synthesis tool PAYNT (Andriushchenko et al. 2025b) as it solves the problems **C2** and **C4**, also on colored MDPs. PAYNT implements three strategies: (1) abstraction-refinement (AR), (2) counterexample-guided inductive synthesis (CEGIS) and (3) Hybrid, combining AR and CEGIS. These strategies have been optimized

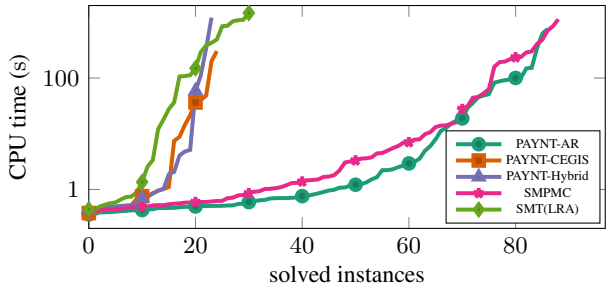


Figure 7: Cactus Plot for **C4**. A method's line going through a point (x, y) means that the method solved the xth benchmark, sorted by time, within y seconds (log-scale).

towards the benchmarks in **C4**. `PAYNT-CEGIS` was specifically developed for problems in **C2**. We have mildly extended `PAYNT-AR` to support the problems in **C3** by adding an inner (non-naive) 1-by-1 evaluation. See Appendix D.3 for details and a comparison with an alternative `PAYNT-AR` extension. Some dedicated approaches support subsets or variations of the problems defined in the classes **C2**-**C4** but the differences prevent a direct performance comparison. We discuss these approaches in the related work.

C2: Constrained Synthesis. Our benchmarks include two application domains: 1) In a set of MCs, find an MC that satisfies a given bound on the implementation cost (Češka et al. 2021). 2) For a given POMDP, find a DT with n nodes encoding a satisfying memoryless policy. We again consider $n \in \{1, \ldots, 15\}$. This benchmark set includes, e.g., the *"refuel-06"* POMDP where the minimal decision tree has 11 nodes. Table 2 shows that SMPMC is on par with `PAYNT-CEGIS` and the `SMT(LRA)` baseline significantly lags behind. The results in Fig. 9 (Appendix D.2) show that there is no clear winner in terms of runtimes.

C3: Robust Synthesis. These benchmarks include synthesis of robust (finite-state) policies in ME-(PO)MDPs. Additionally, we consider sets of MCs with for-all quantified variables. Table 2 shows that SMPMC is on par with the extended version `PAYNT-AR`: Both methods solve the same number of benchmarks. Interestingly, there are 16 benchmarks solved only by SMPMC and 16 different benchmarks solved only by `PAYNT-AR`. Results in Fig. 10 indicate that on the benchmarks solved by both tools, SMPMC is generally faster. `SMT(LRA)` again significantly lags behind.

C4: Plain Synthesis. We consider benchmarks including (1) synthesis of a satisfying fixed-memory FSCs and 2) synthesis of a satisfying MC in a given set of MCs. We also include benchmarks with over 1 million states to showcase that the size of the state space on its own does not pose problems for SMPMC. Table 2 shows that SMPMC is on par with `PAYNT-AR`, which is currently the fastest algorithm for these problems (Andriushchenko et al. 2025b): SMPMC solves two more benchmarks, but `PAYNT-AR` has more unique solutions. Additionally, the results in Fig. 7 show that SMPMC is competitive with `PAYNT-AR` also in terms of runtime. The `SMT(LRA)`, `PAYNT-CEGIS` and `PAYNT-Hybrid` lag significantly behind.

Summary Q2. The results obtained for the synthesis problems in **C2**-**C4** demonstrate that SMPMC is highly competitive with state-of-the-art tools for various widely studied fragments of the problem statement. In contrast to these tools, SMPMC does not sacrifice generality and achieves efficiency without implementing any heuristics that would leverage the structure of the subproblems or additional information from model checking, which opens an interesting avenue for improving the performance of our approach.

6 Related Work

Closest to this work are the approaches in PAYNT, see the introduction and the experimental evaluation for details.

SMT and MILPs. This paper integrates SAT with PMC and thus stands in a tradition of approaches that integrate SAT-style reasoning with dedicated engines, such as mixed-integer (linear) programming (Hooker 2004; Achterberg 2007). CDCL(T)-based approaches like ours exist for theories ranging from convex optimization (Shoukry et al. 2017) to logics on Kripke structures (Eisenhofer et al. 2023). Planning Modulo Theories integrates planning tasks into SMT, enabling the use of theories (Bofill, Espasa, and Villaret 2017). This paper leverages MBQI for quantifiers, whereas another quantifier instantiation strategy is the syntax-based E-matching (de Moura and Bjørner 2007). MBQI has recently been integrated with syntax-based strategies (Kondylidou, Reynolds, and Blanchette 2025).

Controllers for (ME-)POMDPs. There exist several orthogonal approaches for controller synthesis in POMDPs, including various (monolithic) MILP formulations (Amato, Bonet, and Zilberstein 2010; Kumar and Zilberstein 2015b) or analysing the belief space (Kurniawati, Hsu, and Lee 2008; Ho et al. 2024). Such methods can be effectively integrated with FSC synthesis (Andriushchenko et al. 2023). Recently, a lot of focus has been given to finding robust controllers for multiple-environment (PO-)MDPs including sequential convex programming (Cubuktepe et al. 2021), value iteration methods (Nakao, Jiang, and Shen 2021), belief space analysis (Chatterjee et al. 2020) or policy optimisation (Lin et al. 2024; Galesloot et al. 2025).

DT policies for MDPs. The existing tools for finding DTs either construct a DT from a given (optimal) policy (Ashok et al. 2021) or consider a formulation in which only minimality with respect to the depth of the tree is encoded (Vos and Verwer 2023; Andriushchenko et al. 2025a). The DT encoding considered in this work was inspired by the encoding using propositional formulas (Narodytska et al. 2018). Outside of the world of MDPs, finding minimal-sized DTs from training data has been explored (Staus et al. 2025).

7 Conclusion

This paper presents SMPMC: *satisfiability modulo probabilistic model checking*, which tightly integrates the power of satisfiability solvers and model checkers. Among various applications, this approach is the first effective method to find decision tree policies that robustly solve a set of MDPs.

Acknowledgements

The authors are grateful to Clemens Eisenhofer for his help with Z3 and its user propagator API. This work has been executed under the project VASSAL: "Verification and Analysis for Safety and Security of Applications in Life" funded by the European Union under Horizon Europe WIDERA Coordination and Support Action/Grant Agreement No. 10116002. This work has been funded by the NWO VENI Grant ProMiSe (222.147) and the Czech Science Foundation grant GA23-06963S (VESCAA).

References

Achterberg, T. 2007. Conflict analysis in mixed integer programming. *Discret. Optim.*, 4(1): 4–20.

Amato, C.; Bonet, B.; and Zilberstein, S. 2010. Finite-State Controllers Based on Mealy Machines for Centralized and Decentralized POMDPs. In *AAAI*, 1052–1058. AAAI Press.

Andriushchenko, R.; Bork, A.; Češka, M.; Junges, S.; Katoen, J-P.; and Macák, F. 2023. Search and Explore: Symbiotic Policy Synthesis in POMDPs. In *CAV*. ISBN 978-3-031-37709-9.

Andriushchenko, R.; Češka, M.; Junges, S.; and Macák, F. 2024. Policies Grow on Trees: Model Checking Families of MDPs. In *ATVA (2)*, volume 15055 of *LNCS*, 51–75. Springer.

Andriushchenko, R.; Češka, M.; Junges, S.; and Macák, F. 2025a. Small Decision Trees for MDPs with Deductive Synthesis. In *CAV*, 169–192. Springer.

Andriushchenko, R.; Češka, M.; Macák, F.; Junges, S.; and Katoen, J-P. 2025b. An Oracle-Guided Approach to Constrained Policy Synthesis Under Uncertainty. *J. Artif. Intell. Res.*, 82: 433–469.

Ashok, P.; Jackermeier, M.; Křetínský, J.; Weinhuber, C.; Weininger, M.; and Yadav, M. 2021. dtControl 2.0: Explainable Strategy Representation via Decision Tree Learning Steered by Experts. In *TACAS*.

Bayardo Jr., R. J.; and Schrag, R. 1997. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In *AAAI/IAAI*, 203–208. AAAI Press / The MIT Press.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Bjørner, N. S.; Eisenhofer, C.; and Kovács, L. 2023. Satisfiability Modulo Custom Theories in Z3. In *VMCAI*, volume 13881 of *LNCS*, 91–105. Springer.

Bofill, M.; Espasa, J.; and Villaret, M. 2017. Relaxed Exists-Step Plans in Planning as SMT. In *IJCAI*, 563–570. ijcai.org.

Češka, M.; Hensel, C.; Junges, S.; and Katoen, J-P. 2021. Counterexample-guided inductive synthesis for probabilistic systems. *Formal Aspects of Computing*, 33(4): 637–667.

Chades, I.; Carwardine, J.; Martin, T. G.; Nicol, S.; Sabbadin, R.; and Buffet, O. 2012. MOMDPs: A Solution for Modelling Adaptive Management Problems. In *AAAI*. AAAI Press.

Chatterjee, K.; Chmelík, M.; Karkhanis, D.; Novotný, P.; and Royer, A. 2020. Multiple-Environment Markov Decision Processes: Efficient Analysis and Applications. In *ICAPS*, 48–56. AAAI Press.

Chatterjee, K.; Doyen, L.; and Henzinger, T. A. 2010. Qualitative Analysis of Partially-Observable Markov Decision Processes. In *MFCS*, volume 6281 of *Lecture Notes in Computer Science*, 258–269. Springer.

Chrszon, P.; Dubslaff, C.; Klüppelholz, S.; and Baier, C. 2018. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Aspects Comput.*, 30(1): 45–75.

Cubuktepe, M.; Jansen, N.; Junges, S.; Marandi, A.; Suilen, M.; and Topcu, U. 2021. Robust Finite-State Controllers for Uncertain POMDPs. In *AAAI*, 11792–11800. AAAI Press.

de Moura, L. M.; and Bjørner, N. S. 2007. Efficient E-Matching for SMT Solvers. In *CADE*, volume 4603 of *Lecture Notes in Computer Science*, 183–198. Springer.

de Moura, L. M.; and Bjørner, N. S. 2008. Z3: An Efficient SMT Solver. In *TACAS*, volume 4963 of *LNCS*, 337–340. Springer.

Eisenhofer, C.; Alassaf, R.; Rawson, M.; and Kovács, L. 2023. Non-Classical Logics in Satisfiability Modulo Theories. In *TABLEAUX*, volume 14278 of *Lecture Notes in Computer Science*, 24–36. Springer.

Galesloot, M. F. L.; Andriushchenko, R.; Ceska, M.; Junges, S.; and Jansen, N. 2025. Robust Finite-Memory Policy Gradients for Hidden-Model POMDPs. In *IJCAI*, 8518–8526. ijcai.org.

Ge, Y.; and de Moura, L. M. 2009. Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories. In *CAV*, volume 5643 of *LNCS*, 306–320. Springer.

Ghezzi, C.; and Sharifloo, A. M. 2013. Model-based verification of quantitative non-functional properties for software product lines. *Inf. Softw. Technol.*, 55(3): 508–524.

Hartmanns, A.; Junges, S.; Quatmann, T.; and Weininger, M. 2023. A Practitioner's Guide to MDP Model Checking Algorithms. In *TACAS (1)*, volume 13993 of *LNCS*, 469–488. Springer.

Hartmanns, A.; and Kaminski, B. L. 2020. Optimistic Value Iteration. In *CAV (2)*, volume 12225 of *LNCS*, 488–511. Springer.

Hensel, C.; Junges, S.; Katoen, J-P.; Quatmann, T.; and Volk, M. 2022. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4): 589–610.

Ho, Q. H.; Feather, M. S.; Rossi, F.; Sunberg, Z.; and Lahijanian, M. 2024. Sound Heuristic Search Value Iteration for Undiscounted POMDPs with Reachability Objectives. In *UAI*, volume 244 of *Proceedings of Machine Learning Research*, 1681–1697. PMLR.

Hooker, J. N. 2004. A Hybrid Method for Planning and Scheduling. In Wallace, M., ed., *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, 305–316. Springer.

Kondylidou, L.; Reynolds, A.; and Blanchette, J. 2025. Augmenting Model-Based Instantiation with Fast Enumeration. In *TACAS*, 85–103. Springer. ISBN 978-3-031-90643-5.

Kumar, A.; and Zilberstein, S. 2015a. History-Based Controller Design and Optimization for Partially Observable MDPs. In *ICAPS*, 156–164. AAAI Press.

Kumar, A.; and Zilberstein, S. 2015b. History-based controller design and optimization for partially observable MDPs. In *ICAPS*, volume 25, 156–164.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems IV*. The MIT Press.

Lin, Z.; Xue, C.; Deng, Q.; and Ye, Y. 2024. A Single-Loop Robust Policy Gradient Method for Robust Markov Decision Processes. In *ICML*. OpenReview.net.

Nakao, H.; Jiang, R.; and Shen, S. 2021. Distributionally Robust Partially Observable Markov Decision Process with Moment-Based Ambiguity. *SIAM J. Optim.*, 31(1): 461–488.

Narodytska, N.; Ignatiev, A.; Pereira, F.; and Marques-Silva, J. 2018. Learning Optimal Decision Trees with SAT. In *IJCAI*, 1362–1368. ijcai.org.

Nieuwenhuis, R.; Oliveras, A.; and Tinelli, C. 2006. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM*, 53(6): 937–977.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

Shoukry, Y.; Nuzzo, P.; Sangiovanni-Vincentelli, A. L.; Seshia, S. A.; Pappas, G. J.; and Tabuada, P. 2017. SMC: Satisfiability Modulo Convex Optimization. In *HSCC*, 19–28. ACM.

Silva, J. P. M.; and Sakallah, K. A. 1999. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Computers*, 48(5): 506–521.

Smallwood, R. D.; and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Oper. Res.*, 21(5): 1071–1088.

Staus, L. P.; Komusiewicz, C.; Sommer, F.; and Sorge, M. 2025. Witty: An Efficient Solver for Computing Minimum-Size Decision Trees. In *AAAI*, 20584–20591. AAAI Press.

van der Vegt, M.; Jansen, N.; and Junges, S. 2023. Robust Almost-Sure Reachability in Multi-Environment MDPs. In *TACAS (1)*, volume 13993 of *LNCS*, 508–526. Springer.

Vos, D.; and Verwer, S. 2023. Optimal Decision Tree Policies for Markov Decision Processes. In *IJCAI*, 5457–5465. ijcai.org.