Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Saturated Path-Constrained MDP: Planning Under Uncertainty and Deterministic Model-Checking Constraints

Jonathan Sprauel and **Florent Teichteil-Königsbuch**
{jonathan.sprauel, florent.teichteil}@onera.fr
ONERA – The French Aerospace Lab
2 Avenue Édouard-Belin, F-31055 Toulouse, France

Andrey Kolobov
akolobov@microsoft.com
Microsoft Research
Redmond, WA-98052, USA

Abstract

In many probabilistic planning scenarios, a system's behavior needs to not only maximize the expected utility but also obey certain restrictions. This paper presents Saturated Path-Constrained Markov Decision Processes (SPC MDPs), a new MDP type for planning under uncertainty with deterministic model-checking constraints, e.g., "state s must be visited before s'", "the system must end up in s", or "the system must never enter s". We present a mathematical analysis of SPC MDPs, showing that although SPC MDPs generally have no optimal policies, every instance of this class has an ϵ-optimal randomized policy for any $\epsilon > 0$. We propose a dynamic programming-based algorithm for finding such policies, and empirically demonstrate this algorithm to be orders of magnitude faster than its next-best alternative.

Introduction

Markov Decision Processes (MDPs) are some of the most popular models for optimizing the behavior of stochastic discrete-time dynamical systems. In many MDP applications, the system's behavior, formally called a *policy*, needs to not only maximize the expected utility but also obey certain constraints. For instance, to operate a nuclear power plant one always wants a policy that keeps the system in "safe" states, so that even in case of disruptive exogenous events such as earthquakes it can be shut down with probability 1. Requirements like these have prompted researchers to consider a spectrum of MDP types that allow an explicit specification of various constraint structures. On one end of the spectrum are models where utility is optimized under a set of very simple constraints, e.g., goal states in which the system must end up, as in Stochastic Shortest-Path (SSP) MDPs (Bertsekas 1995). On the other end are formalisms from the controller synthesis literature (Baier et al. 2004) that equip MDPs with languages for expressing sophisticated model-checking constraints, but aim at finding *any* constraint-satisfying solution for an MDP and don't reason about solution utility at all. Hardly any model handles both utility maximization and constraint satisfaction at once.

In this context, we propose a new problem class, Saturated Path-Constrained Markov Decision Processes (SPC MDPs), specifically designed to marry decision-theoretic planning to model-checking. As their name implies, SPC MDPs work with a variety of deterministic constraints on policy execution paths, specifically those of the forms "the solution policy must always visit state s before s'", "the solution policy must eventually visit state s", and "the solution policy must never visit state s". Such constraints can describe goal-oriented scenarios, danger-avoidance objectives, and others. SPC MDPs' constraints are stated in a subset of a model-checking temporal logic. In MDP theory, temporal logics have been previously used in constrained utility-free controller synthesis (Baier et al. 2004) and control knowledge formalization (Gretton, Price, and Thiébaux 2004). To our knowledge, the closest existing class to SPC MDPs is Path-Constrained MDPs (Teichteil-Konigsbuch 2012). Although PC MDPs are more general than SPC MDPs (they allow *probabilistic* – or non-saturated – constraints of the kind "the solution policy must visit state s before s' *at least with probability* p"), their expressiveness comes at a cost — the algorithm for solving them is much less efficient than the one we present for SPC MDPs, as our empirical evaluation shows.

Seemingly, SPC MDPs' constraint types can be handled by simple techniques, e.g., pruning actions that lead to forbidden states. However, when coupled with the reward maximization requirement they give SPC MDPs intricate mathematical properties that require much more sophisticated approaches. The SPC instance in Figure 1 showcases these subtleties. It has two states, A and F; A has a reward-collecting action a_1, with a reward of 1, and an exit action a_2 that leads to F. The system starts in A, and we impose the constraint that the system must eventually go to F. The dis-

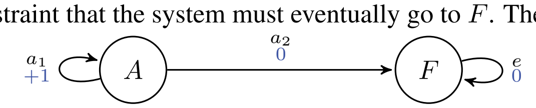


Figure 1: An SPC MDP example

count factor is set to $\gamma = 0.9$. Note that in this MDP there is only one constraint-satisfying Markovian deterministic policy, the one that chooses action a_2 in state A. However, this policy is not the optimal constraint-satisfying solution — randomized and history-dependent policies that repeat a_1 in A $i > 1$ times and then take action a_2 to go to F are valid and earn more reward. This contrasts with existing MDP classes such as infinite-horizon discounted-reward and SSP MDPs, in which at least one optimal policy is guaranteed to be deterministic and history-independent. Since algorithms for these MDP classes are geared towards finding such policies, they all break down on SPC MDPs. Thus, this paper makes the following contributions:

- We define a new class of MDPs, the Saturated Path-Constrained MDPs. We prove that if its set of constraints is feasible, an SPC MDP always has a (possibly randomized) ϵ-optimal policy for any arbitrarily small $\epsilon > 0$, even though an (0-)optimal policy for it may not exist.
- We propose a dynamic programming algorithm that finds an ϵ-optimal policy for any SPC MDP with a feasible constraint set and any $\epsilon > 0$. Its experimental evaluation on a number of benchmarks shows that it is significantly more efficient than the next-best alternative — the fastest algorithm known for PC MDPs (Teichteil-Konigsbuch 2012).

Background

Markov Decision Processes. SPC MDPs are formulated by adding constraints to Infinite-Horizon Discounted-Reward (IHDR) MDPs (Puterman 1994). IHDR MDPs are tuples $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, with state set $\mathcal{S}$, action set $\mathcal{A}$, reward function $\mathcal{R}(s, a)$ giving reward for action a in state s, transition function $\mathcal{T}(s, a, s')$ giving the probability of reaching state s' when taking action a in s, and discount factor $\gamma \in [0, 1)$. In this paper, we assume a probabilistic planning (as opposed to reinforcement learning) setting, where all MDP components are known.

Optimally solving an IHDR MDP means finding a policy π (generally, a mapping from execution histories to distributions over actions) with maximum *value function* $V^\pi(s)$ — the long-term expected utility of following π starting at any initial state s. Any utility-maximizing policy π^* is associated with the *optimal value function* $V^* = \max_\pi V^\pi$. For all $s \in \mathcal{S}$ in an IHDR MDP, $V^*(s)$ exists, is finite, and satisfies the *Bellman equation*:

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{A}} \mathcal{T}(s, a, s') V^*(s') \right\}$$

For IHDR MDP, at least one π^* is *deterministic Markovian*, i.e., has the form $\pi : \mathcal{S} \to \mathcal{A}$. However, for reasons we explain shortly, for SPC MDPs we will look for *randomized* Markovian policies $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

Stochastic Model-Checking. To state SPC MDP policy constraints, we will use a subset of the *Probabilistic Real Time Computation Tree Logic* (PCTL) (Hansson and Jonsson 1994). PCTL allows specifying logical formulas over policy transition graphs using logical operators and boolean functions $f : \mathcal{S} \to \{true, false\}$ on the state space. PCTL constraints are expressed with the *probabilistic "strong until"* temporal operator $fU^{\leq H}_{\diamond p} g$, where $\diamond$ stands for a comparison operator ($<, \leq, =, \geq$, or $>$), meaning that for a randomly sampled execution path of a given policy, the formula $fU^{\leq H} g$ must hold with probability at least/most/equal to p. The formula itself mandates that a given boolean function f must be true for the execution path's states until another boolean function, g, becomes true (f can, but doesn't have to, remain true afterwards). Moreover, g must become true at most H steps after the start of policy execution. If H, called the constraint's *horizon*, equals infinity, g must become true at *some* point on the execution path. A few examples of PCTL constraints in which we are interested in this paper are (initial state s_0 is implicit):

- $(true)U^{\infty}_{=1} g$: *every* policy execution path must eventually visit a state where g holds (mandatory states constraints)
- $(true)U^{\infty}_{=0} g$: *no* policy execution path must ever reach a state where g holds (forbidden states constraints)
- $(\neg f)U^{\infty}_{=0} g$: *no* policy execution path is allowed to visit a state where g holds until it visits a state where f holds (precedence constraints)

Saturated Path-Constrained MDPs

The main contribution of this paper, the Saturated Path-Constrained MDP class, in essense comprises IHDR problems with PCTL constraints in which (1) the horizon H in the "strong until" operator is infinite, i.e. a solution policy must satisfy the constraints at some point, but by no particular deadline, and (2) the probability p in the "strong until" operator equals either 0 or 1, meaning that all constraints are deterministic. Formally, we define SPC MDPs as follows:

Definition 1 (SPC MDP). *A Saturated Path-Constrained MDP (SPC MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, s_0, \gamma, \xi \rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}$, and γ are as in the definition of infinite-horizon discounted-reward MDPs, s_0 is an initial state, and $\xi = \{\xi_1, \ldots, \xi_n\}$ is a set of PCTL "strong until" constraints. For each $i \in [1, n]$, $\xi_i = f_i U^{\infty}_{=p} g_i$, where $f_i, g_i : \mathcal{S} \to \{true, false\}$ are boolean functions and $p = 0$ or 1.*

An SPC MDP policy π is called valid *if it satisfies all the constraints of that MDP for all execution paths starting from the MDP's initial state s_0. An* optimal solution *of an SPC MDP is a policy π that is valid and satisfies:*

$$\forall \textit{ valid } \pi', V^\pi(s_0) \geq V^{\pi'}(s_0).$$

The definition of SPC MDPs might raise a question whether problems with infinite-horizon constraints would be easier to solve if their horizon was approximated by a large but finite number. We believe the answer to be "no", because all known algorithms for finite-horizon MDPs scale exponentially in the horizon, making them impractical for large horizon values. In contrast, algorithms for satisfying infinite-horizon goal-achievement constraints such as those in SPC MDPs and SSP MDPs, do not suffer from this issue.

Importantly, although the horizon of SPC MDPs' constraints is infinite, SPC MDPs can nonetheless handle cases where a goal needs to be attained by a finite deadline. This can be done by introducing a discrete time variable for that goal into the state space and specifying a duration for each action. The advantage of this approach, as opposed to allowing finite horizons in constraints, is that constraint horizons express the maximum *number of actions* to be executed before reaching a goal, not the *amount of real-world time* to do so. Real-world time granularity for achieving different goals may vary, and handling finite-deadline constraints as just described allows for modeling this fact.

Next, we explore SPC MDPs' relationships to existing MDPs and illustrate their expressiveness with an example.

Related models. PC MDPs (Teichteil-Konigsbuch 2012) are the closest existing class to SPC MDPs, and serve as the main point of comparison for the latter in this paper. PC MDPs differ from SPC MDPs only in allowing arbitrary probabilities $p \in [0, 1]$ in their PCTL constraints. Although PC MDPs are a superclass of SPC MDPs, their generality makes them computationally much more demanding. The most efficient known algorithm for finding a near-optimal policy for a PC MDP whenever such a policy exists operates by solving a sequence of linear programs (Teichteil-Konigsbuch 2012), whose length has no known bound. SPC MDPs are an attempt to alleviate PC MDPs' computational issues while retaining much of PC MDPs' expressiveness.

Besides the work on PC MDPs, several other pieces of literature focus on solving MDPs under various kinds of constraints. In Constrained MDPs (Altman 1999), the constraints are imposed on the policy cost. The works of (Etessami et al. 2007; Baier et al. 2004; Younes and Simmons 2004), and (Younes, Musliner, and Simmons 2003) are closer in spirit to SPC MDPs: they use temporal logics to impose restrictions on policy execution paths. In the latter two, the constraints take the form of temporally extended goals. The key difference of all these frameworks from ours is that they ignore policy utility optimization and focus entirely on the constraint satisfaction aspect, whereas SPC MDPs combine these objectives. Other notable models that, like SPC MDPs, consider policy optimization under temporal-logic constraints are NMRDP (Gretton, Price, and Thiébaux 2004; Thiébaux et al. 2006) and work by Kwiatkowska and Parker (2013). The former uses constraints to encode prior control knowledge, but does not handle goals. The latter uses constraints in probabilistic LTL, much like PC MDPs, but does not discount utility and does not handle infinite-reward loops. In the meantime, it is the ability to respect goals and other constraints during discounted utility optimization that is largely responsible for SPC MDPs' mathematical properties.

Temporal logics have been extensively used in combination with MDPs not only to specify constraints but also to characterize non-Markovian rewards (Gretton, Price, and Thiébaux 2004; Thiébaux et al. 2006; Bacchus, Boutilier, and Grove 1997; 1996). While this is a largely orthogonal research direction, its approaches use state space augmentation in ways related to ours. However, due to model specifics, the details are different in each case.

Finally, the following theorem, which is proved in a separate note (Sprauel, Kolobov, and Teichteil-Königsbuch 2014), formally clarifies the inclusion relations between SPC MDPs and several other MDP types.

Theorem 1. *1. The SPC MDP class properly contains the IHDR MDP class and is properly contained by the PC MDP class.*

2. Consider the set of all SPC MDP instances for which there exists a finite $M > 0$ s.t. the value of every policy at every state is well-defined and bounded by M from above for $\gamma = 1$. The set of such SPC MDPs properly contains the GSSP (Kolobov et al. 2011), SSP (Bertsekas 1995), and FH MDP (Puterman 1994) classes.

Example SPC MDP Scenario

As an example of a scenario that can be handled by SPC MDPs, consider designing flight plans for automated drones that monitor forest fires. The drones must plan their missions so as to observe a combination of hard and soft constraints, e.g., surveying zones with high forest fire incidence rates, reckoning areas of extinguished fire, occasionally checking upon zones of secondary importance, and avoiding unauthorized regions such as residential areas.

While soft constraints can be modeled by assigning high rewards, hard constraints present a more serious issue. Indeed, no existing MDP type admits non-terminal goal states, dead-end states from which constraints cannot be met, and arbitrary positive and negative rewards at the same time.

To handle the drone domain with SPC MDPs, we formulate it as a grid where the drone can move in four directions (Figure 2a). We have four possible zone types: (1) high-risk areas (red) with frequent fires that *must* be surveyed at least every x hours, (2) lower-risk areas (orange) where a fire might start and that we should patrol if possible once in a while, (3) mandatory objectives (blue) to be visited occasionally such as sites of previous, extinguished fires and (4) unauthorized zones (black) that must never be entered. We obtain the probability of fire occurrences from multiple statistical sources (Preisler et al. 2004). Once a drone surveys a risky area, the probability of fire there drops to 0 for the next y hours (since, if the drone detects imminent fire danger, this danger is dealt with before fire can fully erupt). When a fire occurs, we assume it is extinguished quickly and another fire can happen at the same location almost immediately.

While we model constraints related to zone type (2) with rewards, as in conventional MDPs, and handle constraints related to zone (1) by augmenting the state space with time variables, as described in the previous section, SPC MDPs allow us to enforce requirements for the other zone types with hard PCTL constraints. For zones of type (3), we use constraints of the form $trueU^{\infty}_{=1}g$. They say that eventually, the drone must visit the states where $g(s) = true$, the function g being *true* only for states in the corresponding zones. For type-(4) zones, we use the constraints $trueU^{\infty}_{=0}g'$, meaning that the drone should never enter states where g' (the indicator for forbidden zones) holds. Note that there is a critical difference between constraints for mandatory objectives in SPC MDPs and goal states in, for example, SSP MDPs — after achieving an SSP MDP goal, the system earns no more reward, while after satisfying a constraint in an SPC

MDP it is free to do so. Note also that we cannot model constraints on forbidden states by simply assigning $-\infty$ for visiting them because then IHDR MDP algorithms would never halt in cases where forbidden states are unavoidable.

Solutions of SPC MDPs

The MDP in Figure 1, described in the Introduction section, illustrates the difficulties of defining and finding optimal solutions for SPC instances. As a reminder, in this MDP an agent starts in state A, and the constraint requires that the agent eventually go to state F. In PCTL, this constraint is expressed as $(true)U^{\infty}_{=1}\mathbb{I}_{s=F}(s)$, where $\mathbb{I}_{s=F}(s)$ is an indicator function that has value *true* iff the current state is F. This example shows two major distinctions of SPC MDPs from conventional MDP classes such as IHDR or SSP:

- *In SPC MDPs, no deterministic Markovian policy may be optimal.* In fact, in Figure 1, the best (and the only) valid such policy, π_{det}, which recommends action a_2 in state A, has the lowest possible reward for this MDP.
- *In SPC MDPs, no optimal policy may exist at all.* For the valid deterministic history-dependent policies $\pi^i_{d.h.}$ in Figure 1, where policy $\pi^i_{d.h.}$ repeats action a_1 in A i times and then takes action a_2, the higher the i, the higher the policy reward. Similarly, for randomized Markovian policies $\pi_{r.M.}$, policy reward increases as $\pi_{r.M.}(A, a_2) \to 0$.

However, this example also gives us intuitions about the nature of SPC MDP solutions that we may want to find:

1) The values of some valid policies (Definition 1) can't be exceeded by much. In the above example, policies that loop in state A for a very long time and then go to F have values approaching $\frac{1}{1-\gamma}$, i.e., are "almost optimal". The following definition formalizes this concept:

Definition 2 (Epsilon-optimality). *A policy π is an ϵ-*optimal solution (Puterman 1994) of a SPC MDP *if π is valid and for all other valid policies π',*

$$V^{\pi}(s_0) \geq V^{\pi'}(s_0) - \epsilon$$

2) The best valid policies for an SPC MDP "resemble" certain deterministic Markovian ones. In the example in Figure 1, the best deterministic policy (but invalid) policy loops in state A for *infinitely* long, while ϵ-optimal valid policies loop in A for *very* long before transitioning to F.

In the following sections, we prove that these intuitions are correct for SPC MDPs in general: for every $\epsilon > 0$, every SPC MDP has a valid randomized ϵ-optimal policy that is similar to a deterministic (but invalid) one for that MDP.

Value Iteration for SPC MDPs

Our algorithm for finding valid ϵ-optimal SPC MDP policies is based on the intuitions above. It has two basic steps: (1) a pruning operation, which compiles away the constraints from the SPC MDP; crucially, this step yields an IHDR MDP for which there exist ϵ-optimal policies that are valid and ϵ-optimal for the original SPC MDP too. (2) a Value Iteration-like procedure, which searches for such special ϵ-optimal policies of this IHDR MDP. The correctness of the resulting algorithm relies on several theorems. **Their proofs are provided in a supplemental document (Sprauel, Kolobov, and Teichteil-Königsbuch 2014).**

We first present the second, Value Iteration-based part, since it is the most important result. As the definitions and theorems below imply, its properties hold for general IHDR MDPs, not just those derived from SPC MDPs. We begin by describing the policies our algorithm will be looking for.

Definition 3 (ω-policies). *Let $A(s)$ be a set of actions in state s of an MDP, $|A(s)| \geq 1$. For an $\omega \in [0, 1]$, an ω-*policy *is a randomized Markovian policy that, in each state s, chooses some action $a \in A(s)$ with probability $(1-\omega)$ (or 1 if a is the only action), and chooses any other action in $A(s)$ with equal probability $\frac{\omega}{|A(s)|-1}$ (or 0 if a is the only action).*

ω-policies relate to Intuition #2 from the previous section. Namely, when ω is small, an ω-policy behaves similarly to a deterministic policy. As Intuition #2 implies and we show shortly, it makes sense to look for ϵ-optimal solutions among ω-policies. Before deriving that result, we demonstrate that the best ω-policy for a given ω can be found easily:

Definition 4 (Bellman Operator for ω-policies). *Let V be a value function of an IHDR MDP, s — a state, and $A(s)$ — a set of actions in s, $|A(s)| > 1$. The ω-Bellman operator is defined as $\omega Bell_{A(s)}V(s) =$*

$$\max_{a \in A(s)} \left[(1-\omega) \left(\mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s')V(s') \right) + \sum_{a' \in A(s) \setminus \{a\}} \frac{\omega \left(\mathcal{R}(s,a') + \gamma \sum_{s'} \mathcal{T}(s,a',s')V(s') \right)}{|A(s)|-1} \right]$$

For states s s.t. $|A(s)| = 1$, the ω-Bellman operator reduces to the classical Bellman operator. It can also be shown that:

Theorem 2 (Convergence of the ω-Bellman Operator.). *For any fixed $\omega \in [0, 1]$, the ω-Bellman Operator applied an infinite number of times to every state of an IHDR MDP has a unique fixed point. This fixed point V^{ω} has the highest value among all ω-policies for that ω.*

We can define an ω-policy associated greedily w.r.t. V^{ω}:

Definition 5 (Greedy ω-policy.). *Let V^{ω} be the fixed point of the ω-Bellman operator for a given ω and IHDR MDP. We call π^{ω} a greedy ω-policy if for each state s there is an action a that maximizes the right-hand-side of ω-Bellman in s, and $\pi^{\omega}(s, a) = 1 - \omega$.*

Finally, the following result in combination with the preceding ones explains why we choose to concentrate on ω-policies in order to find ϵ-optimal solutions:

Theorem 3 (Choosing an ω with ϵ-Optimality Guarantee.). *For a given IHDR MDP, $\gamma < 1$ and any fixed $\epsilon > 0$, choosing $\omega = \frac{\epsilon(1-\gamma)^2}{R_{max}-R_{min}}$ where R_{max} and R_{min} are the maximum and minimum of the reward function, guarantees that any greedy ω-policy is ϵ-optimal.*

Thus, given an ϵ, Theorem 3 proves the existence of an ϵ-optimal ω-policy, and Theorem 2 allows us to find one. The resulting Value Iteration procedure is shown in Algorithm 1.

Recall, however, that the theory above lets us compute an ϵ-optimal solution for an *IHDR* problem, not an SPC MDP. Next, we address this issue by describing a conversion from any SPC MDP to an IHDR instance whose ϵ-optimal solutions are valid and ϵ-optimal for the original problem.

Compiling Away History-Dependent Policies. The example in Figure 1 suggests that taking into account the execution history in SPC MDPs generally leads to improved behavior in the current state. As it turns out, for certain SPC MDPs, history dependence is unnecessary, and at least one ϵ-optimal policy is Markovian for any $\epsilon > 0$. This happens whenever all constraints in the SPC MDP are *transient* (Teichteil-Konigsbuch 2012):

Definition 6 (Transient Constraints). *A constraint encoded by a pair of boolean functions $f, g : \mathcal{S} \rightarrow \{true, false\}$ is* transient *if the three sets of states where $(f(s), g(s)) = (true, true)$, $(false, false)$, $(false, true)$, respectively, are absorbing.*

Fortunately, we can transform problems where some of the constraints are *not* transient into SPC MDPs with only transient constraints.

Theorem 4 (Reduction to Transient Constraints). *Let $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, s_0, \gamma, \xi \rangle$ be an SPC MDP, for which m out of n constraints $\xi = \{\xi_1, \ldots, \xi_n\}$ are non-transient. Then M can be transformed into an SPC MDP $M' = \langle \mathcal{S}', \mathcal{A}, \mathcal{R}', \mathcal{T}', s'_0, \gamma, \xi' \rangle$ with only transient constraints by augmenting its state space with one three-valued ("validated", "invalidated", "unknown") variable per non-transient constraint, and replacing each such constraint with a transient one. M' has $3^m|\mathcal{S}|$ states.*

The first step of our overall algorithm for solving SPC MDPs, shown below, uses these theorems to eliminate non-transient constraints and neutralizes the remaining constraints by deleting states and actions that violate them.

High-Level Algorithm Description. To recapitulate, our algorithm, called SPC Value Iteration (SPC VI), whose main loop is presented in Algorithm 1, operates in two stages:

1. The first step starts by compiling away non-transient constraint functions (Alg. 1, line 2). Then it identifies the states of the given SPC MDP that can't be reached from s_0 without violating at least one constraint (Alg. 2, line 4), the states from which at least one constraint is unsatisfiable, and all actions leading to these types of states (Alg. 3, lines 7 and 9). As Algorithm 3 shows, this can be done with iterative DP. These actions and states aren't part of any valid policy and are removed from the MDP (Alg. 3, line 12), leaving reduced state set $\hat{\mathcal{S}}$ and action sets $A(s)$ for each $s \in \hat{\mathcal{S}}$. These sets form an IHDR MDP with the property described by the following lemma:

   Lemma 1. *For an SPC MDP, let Π_A be the set of all policies on the state set $\hat{\mathcal{S}}$ as defined above, s.t. each $\pi \in \Pi_A$ assigns a 0 probability to every $a \notin A(s)$ in every $s \in \hat{\mathcal{S}}$ it reaches.*

   (a) Π_A contains all valid policies.
   (b) Every ω-policy with $\omega > 0$ in Π_A is valid.

   Thus, the first step outputs pruned sets of the SPC MDP's states and actions to construct any valid policy, *including every valid ω-policy*. I.e., *if there is a valid ϵ-optimal ω-policy*, we must be able to build it from this "kit".

2. The second step (Alg. 1, lines 8-10) begins by determin-

Algorithm 1: SPC MDP Value Iteration

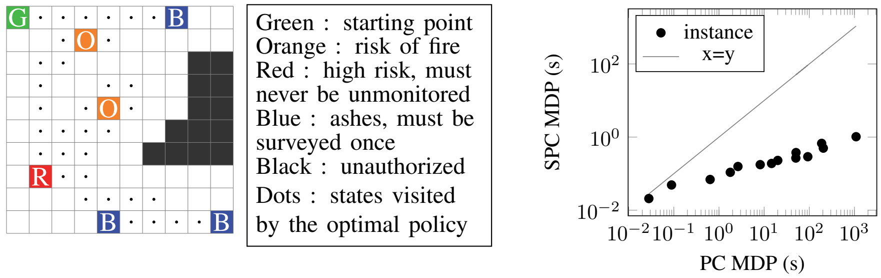
1 **Input:** SPC MDP with constraints $\xi_1, \ldots, \xi_n$, $\epsilon > 0$, $\theta > 0$;
2 Compile away the SPC MDP's non-transient constraints (Theorem 4), if necessary;
3 Initialize the set $\hat{\mathcal{S}} := \{s_0\}$ of explored states;
4 Initialize the set $Tip := \{s_0\}$, a subset of $\hat{\mathcal{S}}$;
5 $explore(\hat{\mathcal{S}}, Tip)$;
6 **for** $i : 1 \ldots n$ **do** $updateReachability(\hat{\mathcal{S}}, \xi_i, \theta)$;
7 **for** $s \in \mathcal{S}$ **do** $A(s) := \left\{a \mid \forall s', T(s, a, s') > 0 \Rightarrow s' \in \hat{\mathcal{S}}\right\}$;
8 Initialize $\omega := \frac{\epsilon(1-\gamma)^2}{R_{max} - R_{min}}$;
9 Compute V^ω by Value Iteration with $\omega Bell$ operator;
10 Return π ω-policy greedy w.r.t V^ω

Algorithm 2: $explore(\hat{\mathcal{S}}, Tip)$ function

1 **repeat**
2 Pick $s \in Tip$;
3 **if** *there is a constraint $\xi_i = f_i U^{\infty}_{=p_i} g_i$ s. t. $((p_i = 1 \,\&\, !f_i(s) \,\&\, !g_i(s))$ or $(p_i = 0 \,\&\, g_i(s)))$* **then**
4 Remove s from Tip and $\hat{\mathcal{S}}$;
5 **else**
6 **for** *a and s' with $T(s, a, s') > 0$* **do**
7 **if** *s' not already in set $\hat{\mathcal{S}}$* **then**
8 Add s' to Tip and $\hat{\mathcal{S}}$
9 **until** $Tip = \emptyset$;

Algorithm 3: $updateReachability(\hat{\mathcal{S}}, \xi_i, \theta)$ function

1 Initialize validity value functions X, X' to 0;
2 **for** $s \in \hat{\mathcal{S}}$ **do** $X(s) := g_i(s)$ s.t. $\xi_i = f_i U^{\infty}_{=p_i} g_i$;
3 **repeat**
4 $X' := X$;
5 **for** $s \in \hat{\mathcal{S}}$ **do**
6 **if** $p_i = 1$ **then**
7 **if** $!f_i(s)$ **then** $X(s) := 0$ **else** $X(s) := \max_a \left[\sum_{s'} \mathcal{T}(s, a, s') X(s')\right]$;
8 **if** $p_i = 0$ **then**
9 **if** $g_i(s)$ **then** $X(s) := 1$ **else** $X(s) := \min_a \left[\sum_{s'} \mathcal{T}(s, a, s') X(s')\right]$;
10 **until** $\mid X - X' \mid_{max} < \theta$;
11 **for** $s \in \hat{\mathcal{S}}$ **do**
12 **if** $((p_i = 1) \,\&\, X(s) < 1 - \theta)$ *or* $((p_i = 0) \,\&\, X(s) > \theta)$ **then** Remove s from $\hat{\mathcal{S}}$ and discard actions leading to it ;



ϵ	N	**t (s)**	**V (discounted)**
10	1	0.045	-19.6931
	2	5.95	-39.1865
	3	54.6	-60.1082
1	1	0.048	-19.1382
	2	5.82	-38.616
	3	53.8	-59.8614
0.1	1	0.041	-19.0826
	2	5.89	-38.5589
	3	54.5	-59.8367

(a) Example of instance of the firefighting domain

(b) Solution times on the firefighting domain (logscale): each dot is an instance.

(c) System update domain

Figure 2: Benchmark results

ing an ω (line 8) for which all ω-policies over the remaining states and actions are ϵ-optimal via Theorem 3, and runs the ω-*Bellman operator* (line 9) to find the greedy ω-policy that is ϵ-optimal for the pruned MDP, and hence, by construction, for the original SPC MDP. By Lemma 1, that policy is also valid, yielding the following result:

Theorem 5 (Existence of ϵ-Optimal ω-Policies.). *For any SPC MDP with satisfiable constraints, there exists a valid ϵ-optimal policy, and SPC VI can find it. The solution is randomized and Markovian in the augmented state space, but history-dependent in the original one.*

Experimental Evaluation

The objectives of our experiments are two-fold: (1) to compare SPC VI with the algorithm for PC MDPs, PC MDP-ILP (Teichteil-Konigsbuch 2012), in terms of efficiency, and (2) to validate SPC MDPs as an efficient modeling tool. The experiments were run with 5.8 GB of RAM on a 2.80GHz CPU. In the experiments, we used two benchmark domains in the PPDDL language, extended to express PCTL constraints. In all experiments, the SPC MDP discount factor was $\gamma = 0.9$. The domains are:

Fire Surveillance Drone Domain. The domain is described in the Example SPC MDP Scenario section. We randomly generated instances with different grid dimensions (between 10x10 and 100x100, with a total number of states between 200 and 160 000), time parameter values, and numbers of zones of each type (between 1 and 5 per type). For ϵ-optimality we set $\epsilon = 0.1$, with a corresponding $\omega = 0.001$, since we fixed the penalty of a fire occurrence to -1.

System Update Domain. The domain involves deploying an important but non-time-critical update on all nodes in a cluster — every node must receive the update eventually. Each node can be updated independently, at a different time. Each node has two power modes, s_1 (low) and s_2 (high). Every hour a node spends in s_1 costs $r_1 = -10$ cents/node, and every hour spent in s_2 costs $r_2 = -20$ cents. Each node can switch between power states once an hour, but each switch costs 30 cents in energy. The update takes 7 hours, during all of which the node must be in the high-power state.

At any point, each node is either in use or not. Every hour a node is not in use, with probability p a user starts utilizing it. If this happens, the node stays busy for the next 4 hours, during which no other users can use it. Crucially, the probability p of a new user appearing changes every 12 hours between 1/4 during daytime and 1/8 during nighttime.

Users pay 50 cents/hour for a node in a high-power state (however, 20 of these go to pay for energy costs) and 10 cents/hour for a node in a low-power state. When a node is getting updated while in use, administrators have to compensate its users with 50 cents/hour for the disruption (in addition to paying for the system being in a high-power state).

To summarize, for every hour (time step) there are three actions per node: (1) update, (2) switch its power state, (3) do nothing. The constraint is that every node must get updated eventually and must be running in a high-power state for the update's duration. Since the reward function ranges from -50 to 50, we chose a parameter $\epsilon \leq 10$ for the ϵ-optimality; the corresponding ω parameter is 0.001. We tested instances having from 1 computer (1246 states) to 3 computers (1 014 013 states).

We used the Drone domain to compare SPC VI against PC MDP-ILP (Teichteil-Konigsbuch 2012); they both produce the same type of solutions (randomized, ϵ-optimal policies, Markovian if all functions are transient). An important motivation for our work has been to formulate an MDP model with constraints that would be more efficient to solve than PC MDPs, with little loss in expressiveness. As Figure 2 shows, SPC MDPs meet this criterion. While the domain can be formulated as either of these MDP types, SPC VI is much faster than PC MDP-ILP across all the domain instances we experimented with (the dots corresponding to different instances are all below the diagonal in the (b) plot).

For the Update domain, we aimed to study the influence of ϵ. As Table 2c shows, there is no obvious effect of choosing a lower ϵ on the total solving time spent. This can mainly be explained by noting that the ϵ in SPC VI is not used for termination (as in VI for IHDR MDP's or indirectly in PC MDP-ILP), so has little influence on SPC VI's runtime.

We also assessed the obtained policies qualitatively, and it turned out that they tended to install the update at night, right after the last "day user" stopped using the node in high-power mode. This is very intuitive: at night, the risk of costly user interference with the update is lower. In addition, by installing the update right after a user quit, when the node is still in a high-power state, the system avoided wasting resources switching to a low-power state and back in order to satisfy the energy requirements of update installation later.

Conclusion

This paper introduced Saturated Path-Constrained MDPs, an MDP model that enables designers to express natural hard constraints on the desired policies, and an algorithm for provably finding ϵ-optimal policies for them. SPC MDPs are strictly more expressive than several existing MDP types, e.g., stochastic shortest-path and discounted-reward formulations. It strikes a good balance between these models and PC MDPs — although the latter are somewhat more expressive, they are significantly harder to solve.

As the next step, we plan to develop even more efficient techniques for solving SPC MDPs, in an effort to bring this framework closer to complex industrial applications.

References

Altman, E. 1999. *Constrained Markov Decision Processes*. Chapman and Hall/CRC.

Bacchus, F.; Boutilier, C.; and Grove, A. 1996. Rewarding behaviors. In *The Thirteenth National Conference on Artificial Intelligence (AAAI)*.

Bacchus, F.; Boutilier, C.; and Grove, A. 1997. Structured solution methods for non-Markovian decision processes. In *The Fourteenth National Conference on Artificial Intelligence (AAAI)*.

Baier, C.; Größer, M.; Leucker, M.; Bollig, B.; and Ciesinski, F. 2004. Controller synthesis for probabilistic systems. In *IFIP TCS*.

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control*. Athena Scientific.

Etessami, K.; Kwiatkowska, M.; Vardi, M. Y.; and Yannakakis, M. 2007. Multi-objective model checking of Markov decision processes. In *Proc. of the 13th int. conf. on Tools and algorithms for the construction and analysis of systems*, 5065. Berlin, Heidelberg: SpringerVerlag.

Gretton, C.; Price, D.; and Thiébaux, S. 2004. Nmrdpp: Decision-theoretic planning with control knowledge. In *In Proceedings of the Probablistic Planning Track of IPC-04*.

Hansson, H., and Jonsson, B. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* (6):512–535.

Kolobov, A.; Mausam, M.; Weld, D.; and Geffner, H. 2011. Heuristic search for generalized stochastic shortest path mdps. *International Conference on Automated Planning and Scheduling*.

Kwiatkowska, M., and Parker, D. 2013. Automated verification and strategy synthesis for probabilistic systems. In Hung, D. V., and Ogawa, M., eds., *Proc. 11th International Symposium on Automated Technology for Verification and Analysis (ATVA'13)*, volume 8172 of *LNCS*, 5–22. Springer.

Preisler, H. K.; Brillinger, D. R.; Burgan, R. E.; and Benoit, J. 2004. Probability based models for estimation of wildfire risk. *International Journal of Wildland Fire* 13(2):133–142.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley and Sons, Inc., 1st edition.

Sprauel, J.; Kolobov, A.; and Teichteil-Königsbuch, F. 2014. Saturated Path-Constrained MDP: Complementary Proofs. http://www.onera.fr/sites/default/files/u701/aaai14_spcmdp_proofs.pdf.

Teichteil-Konigsbuch, F. 2012. Path-constrained Markov decision processes: bridging the gap between probabilistic model-checking and decision-theoretic planning. *The Twentieth European Conference on Artificial Intelligence (ECAI)*.

Thiébaux, S.; Gretton, C.; Slaney, J.; Price, D.; and Kabanza, F. 2006. Decision-theoretic planning with non-Markovian rewards. *Journal Artificial Intelligence Research* 25(1):17–74.

Younes, H. L. S., and Simmons, R. G. 2004. Policy generation for continuous-time stochastic domains with concurrency. In *ICAPS*, 325–334.

Younes, H. L. S.; Musliner, D. J.; and Simmons, R. G. 2003. A framework for planning in continuous-time stochastic domains. In *ICAPS*, 195–204.