Efficient Strategy Synthesis for MDPs with Resource Constraints

František Blahoudek, Petr Novotný, Melkior Ornik, Pranay Thangeda, and Ufuk Topcu

***Abstract*—We consider qualitative strategy synthesis for the formalism called consumption Markov decision processes. This formalism can model the dynamics of an agent that operates under resource constraints in a stochastic environment. The presented algorithms work in time polynomial with respect to the representation of the model and they synthesize strategies ensuring that a given set of goal states will be reached (once or infinitely many times) with probability 1 without resource exhaustion. In particular, when the amount of resource becomes too low to safely continue in the mission, the strategy changes course of the agent towards one of a designated set of reload states where the agent replenishes the resource to full capacity; with a sufficient amount of resource, the agent attempts to fulfill the mission again. We also present two heuristics that attempt to reduce the expected time that the agent needs to fulfill the given mission, a parameter important in practical planning. The presented algorithms were implemented and numerical examples demonstrate (i) the effectiveness (in terms of computation time) of the planning approach based on consumption Markov decision processes and (ii) the positive impact of the two heuristics on planning in a realistic example.**

***Index Terms*—consumption Markov decision process, planning, resource constraints, strategy synthesis**

I. Introduction

Autonomous agents like driverless cars, drones, or planetary rovers typically operate under resource constraints and are often deployed in stochastic environments which exhibit uncertain outcomes of the agents' actions [1]. Markov decision processes (MDPs) are commonly used to model such environments for planning purposes [2]. Intuitively, an MDP is described by a set of states and transitions between these states. In a discrete-time MDP, the evolution happens in discrete steps and a transition has two phases: first, the agent chooses some action to play, and the resulting state is chosen randomly based on a probability distribution defined by the action and the agent's state.

Submitted for review on 26 January, 2021. This work was partially supported by NASA's grant "Safety-Constrained and Efficient Learning for Resilient Autonomous Space Systems", by DARPA's grant HR001120C0065, by ARO's grant W911NF-20-1-0140, by ONR's grant N00014-18-1-2829, and by the Czech Ministry of Education, Youth and Sports project LL1908 of the ERC.CZ programme. Petr Novotný is supported by the Czech Science Foundation Junior grant GA21-24711S.

František Blahoudek was with the Oden Institute, The University of Texas at Austin, Austin, USA (e-mail: frantisek.blahoudek@gmail.com).

Petr Novotný is with the Faculty of Informatics, Masaryk University, Brno, Czech Republic (e-mail: petr.novotny@fi.muni.cz).

Melkior Ornik and Pranay Thangeda are with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, USA (e-mail: mornik@illinois.edu, pranayt2@illinois.edu).

Ufuk Topcu is with the Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, USA (e-mail: utopcu@utexas.edu).

The interaction of an agent with an MDP is formalized using strategies. A strategy is simply a recipe that tells the agent, in every moment, what action to play next. The problem of finding strategies suitable for given objectives is called strategy synthesis for MDPs.

Strategy synthesis with resource constraints in real-world systems is computationally expensive. Introducing a continuous resource state would convert the system either to a stochastic hybrid system – a framework in which control synthesis is, when at all feasible, is often computationally intractable [3]–[6] – or necessitate removing the stochastic element of the problem. On the other hand, adding all possible resource levels to the state space in an MDP leads to an explosion in the size of the discrete state space.

As the main results of this paper, we solve polynomial-time strategy synthesis for the following two objectives in resource-constrained MDPs: (i) almost-sure reachability of a given set of states T, and (ii) almost-sure Büchi objective for T. That is, the synthesized strategies ensure that, with probability 1 and without resource exhaustion, some target from T will be reached at least once or T will be visited infinitely often.

We also present two heuristics that improve the practical utility of the presented algorithms for planning in resource-constrained systems. In particular, the goal-leaning and threshold heuristics attempt, as a secondary objective, to reach T in a short time. Further, we briefly describe our tool implementing these algorithms and we demonstrate that our approach specialized to qualitative analysis of resource-constrained systems can solve this task faster than the state-of-the-art general-purpose probabilistic model checker Storm [7].

A. Current approaches to resource-constraints.

There is a substantial body of work in the area of verification of resource-constrained systems [8]–[16]. A naive approach is to model such systems as finite-state systems with states augmented by an integer variable representing the current resource level. The resource constraint requires that the resource level never drops below zero.

The well-known energy model [8], [9] avoids the encoding of the resource level into state space: instead, the model uses an integer counter, transitions are labeled by integers, and taking an ℓ-labelled transition results in ℓ being added to the counter. Thus, negative numbers stand for resource consumption while positive ones represent charging. Many variants of both MDP and game-based energy models have been studied. In particular, [17] considers strategy synthesis for energy MDPs with qualitative Büchi and parity objectives.

The main limitation of the energy models is that in general, they are not known to admit strategy synthesis algorithms that work in time polynomial with respect to the representation of the model. Indeed, already the simplest problem, deciding whether a non-negative energy can be maintained in a two-player energy game is at least as hard as solving mean-payoff graph games [9]; the question whether the latter belongs to P is a well-known open problem [18]. This hardness translates also to MDPs [17], making polynomial-time strategy synthesis for energy MDPs impossible without a theoretical breakthrough.

B. Consumption MDPs

Our work is centered around Consumption MDPs (CMDPs) which is a model motivated by real-world vehicle energy consumption and inspired by consumption games [19]. In a CMDP, the agent has a finite storage capacity, each action consumes a non-negative amount of resource, and replenishing of the resource happens only in designated states, called reload states. Resource replenishment at reload states occurs as an atomic (instant) event – an assumption that holds true in many real-world settings.

Reloading as atomic events and bounded capacity are the key ingredients for efficient analysis of CMDPs. Our qualitative strategy synthesis algorithms work provably in time that is polynomial with respect to the representation of the model. Moreover, they synthesize strategies with a simple structure and an efficient representation via binary counters.

The notion of CMDPs and the algorithm for the Büchi objective were first introduced in [20]. This paper builds upon [20], substantially expanding it by extending the algorithmic core with the reachability objective (Section VII) and introducing goal-leaning and threshold heuristics to improve expected reachability time of targets (Section VIII). Further, the notation in the paper is completely overhauled to simplify the understanding of the merits and proofs, it uses pictorial examples to improve readability, and also includes complete proofs omitted in [20]. Finally, the numerical experiments (Section IX) section includes examples that uses STORM as a baseline for comparison.

C. Outline

Section II introduces CMDPs with the necessary notation and it is followed by Section III which discusses strategies with binary counters. Sections IV and V solve two intermediate objectives for CMDPs, namely safety and positive reachability, that serve as stepping stones for the main results. The solution for the Büchi objective is conceptually simpler than the one for almost-sure reachability and thus is presented first in Section VI, followed by Section VII for the latter. Section VIII defines expected reachability time and proposes the two heuristics for its reduction. Finally, Section IX describes our implementation and an illustrative example showing the utility of our algorithms. It also provides two numerical experiments showing the effectiveness of CMDPs for analysis of resource-constrained systems and the impact of the proposed heuristics on expected reachability time. For better readability, two rather technical proofs were moved from Section V to Appendix.

II. PRELIMINARIES

We denote by $\mathbb{N}$ the set of all non-negative integers and by $\overline{\mathbb{N}}$ the set $\mathbb{N} \cup \{\infty\}$. For a set I and a vector $\mathbf{v} \in \overline{\mathbb{N}}^I$ indexed by I we use $\mathbf{v}(i)$ for the i-component of $\mathbf{v}$. We assume familiarity with basic notions of probability theory.

A. Consumption Markov decision processes (CMDPs)

Definition 1 (CMDP)**.** *A* consumption Markov decision process *(CMDP) is a tuple* $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ *where* S *is a finite set of* states*,* A *is a finite set of* actions*,* $\Delta\colon S \times A \times S \to [0,1]$ *is a* transition function *such that for all* $s \in S$ *and* $a \in A$ *we have that* $\sum_{t \in S} \Delta(s,a,t) = 1$*,* $\gamma\colon S \times A \to \mathbb{N}$ *is a* consumption function*,* $R \subseteq S$ *is a set of* reload states *where the resource can be reloaded, and* cap *is a* resource capacity*.*

Visual representation. CMDPs are visualized as shown in Fig. 1 for a CMDP $(\{s,t,u,v,w\}, \{\mathsf{a},\mathsf{b}\}, \Delta, \gamma, \{s,u\}, 20)$. States are circles, reload states are double circled, and target states (used later for reachability and Büchi objectives) are highlighted with a green background. Capacity is given in the yellow box. The functions Δ and γ are given by (possibly branching) edges in the graph. Each edge is labeled by the name of the action and by its consumption enclosed in brackets. Probabilities of outcomes are given by gray labels in proximity of the respective successors. To avoid clutter, we omit 1 for non-branching edges and we merge edges that differ only in action names.

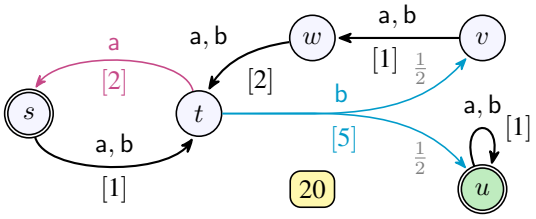


Fig. 1: A CMDP with a target set $\{u\}$.

For $s \in S$ and $a \in A$, we denote by $Succ(s,a)$ the set $\{t \mid \Delta(s,a,t) > 0\}$. A path is a (finite or infinite) state-action sequence $\alpha = s_1 a_1 s_2 a_2 s_3 \cdots \in (S \cdot A)^\omega \cup (S \cdot A)^* \cdot S$ such that $s_{i+1} \in Succ(s_i, a_i)$ for all i. We define $\alpha_i = s_i$ and we say that α is s_1-initiated. We use $\alpha_{..i}$ for the finite prefix $s_1 a_1 \dots s_i$ of α, $\alpha_{i..}$ for the suffix $s_i a_i \dots$, and $\alpha_{i..j}$ for the infix $s_i a_i \dots s_j$. A finite path is a cycle if it starts and ends in the same state and is simple if none of its proper infixes forms a cycle. The length of a path α is the number $len(\alpha)$ of actions on α, with $len(\alpha) = \infty$ if α is infinite.

An infinite path is called a run. We typically name runs by variants of the symbol ϱ. A finite path is called history. We use $last(\alpha)$ for the last state of a history α. For a history α with $last(\alpha) = s_1$ and for $\beta = s_1 a_1 s_2 a_2 \dots$ we define a joint path as $\alpha \odot \beta = \alpha a_1 s_2 a_2 \dots$.

A CMDP is decreasing if for every cycle $s_1 a_1 s_2 \dots a_{k-1} s_k$ there exists $1 \leq i < k$ such that $\gamma(s_i, a_i) > 0$. Throughout this paper we consider only decreasing CMDPs. The only place where this assumption is used are the proofs of Theorem 3 and Theorem 7.

B. Resource: consumption and levels

The semantics of the consumption γ, reload states R and capacity *cap* naturally capture the evolution of levels of the resource along paths in $\mathcal{C}$. Intuitively, each computation of $\mathcal{C}$ must start with some initial load of the resource, actions consume the resource, and reload states replenish the resource level to *cap*. The resource is depleted if its level drops below 0, which we indicate by the symbol $\bot$ in the following.

Formally, let $\alpha = s_1 a_1 s_2 \dots s_n$ (where n might be ∞) be a path in $\mathcal{C}$ and let $0 \leq d \leq cap$ be an initial load. We write ${}^d\alpha$ to denote the fact that α started with d units of the resource. We say that α is loaded with d and that ${}^d\alpha$ is a loaded path. The resource levels of ${}^d\alpha$ is the sequence $RL_{\mathcal{C}}({}^d\alpha) = r_1, r_2, \dots, r_n$ where $r_1 = d$ and for $1 \leq i < n$ the next resource level r_{i+1} is defined inductively, using $c_i = \gamma(s_i, a_i)$ for the consumption of a_i, as

$$r_{i+1} = \begin{cases} r_i - c_i & \text{if } s_i \notin R \text{ and } c_i \leq r_i \neq \bot, \\ cap - c_i & \text{if } s_i \in R \text{ and } c_i \leq cap \text{ and } r_i \neq \bot, \\ \bot & \text{otherwise.} \end{cases} \tag{1}$$

If α (and thus n) is finite, we use $lastRL_{\mathcal{C}}({}^d\alpha)$ to reference the last resource level r_n of ${}^d\alpha$.

A loaded path ${}^d\alpha$ is safe if $\bot$ is not present in $RL_{\mathcal{C}}({}^d\alpha)$, which we write as $\bot \notin RL_{\mathcal{C}}({}^d\alpha)$. Naturally, if ${}^d\alpha$ is safe then ${}^h\alpha$ is safe for all $h \geq d$.

Example 1. *Consider the CMDP in Fig. 1 with capacity* 20 *and the run* $\varrho = (t\mathsf{a}s\mathsf{a})^\omega$ *with the initial load* 2*. We have that* $RL_{\mathcal{C}}({}^2\varrho) = 2, 0, 19, 17, 19, 17 \dots$ *and thus* ${}^2\varrho$ *is safe. On the other hand, for the run* $\varrho' = (t\mathsf{b}u\mathsf{a}w\mathsf{a})^\omega$ *we have* $RL_{\mathcal{C}}({}^{20}\varrho') = 20, 15, 14, 12, 7, 6, 4, \bot, \bot, \dots$ *and, in fact, no initial load can make* ϱ' *safe.*

C. Strategies

A strategy σ for $\mathcal{C}$ is a function assigning an action to each loaded history. An evolution of $\mathcal{C}$ under the control of σ starting in some initial state $s \in S$ with an initial load $d \leq cap$ creates a loaded path ${}^d\alpha = {}^d s_1 a_1 s_2 \dots$ as follows. The path starts with $s_1 = s$ and for $i \geq 1$ the action a_i is selected by the strategy as $a_i = \sigma({}^d s_1 a_1 s_2 \dots s_i)$, and the next state s_{i+1} is chosen randomly according to the values of $\Delta(s_i, a_i, \cdot)$. Repeating this process ad infinitum yields an infinite sample run (loaded by d). Loaded runs created by this process are σ-compatible. We denote the set of all σ-compatible s-initiated runs loaded by d by $\mathsf{Comp}_{\mathcal{C}}(\sigma, s, d)$.

We denote by ${}^d_s\mathbb{P}^\sigma_{\mathcal{C}}(\mathsf{A})$ the probability that a sample run from $\mathsf{Comp}_{\mathcal{C}}(\sigma, s, d)$ belongs to a given measurable set of loaded runs A. For details on the formal construction of measurable sets of runs see [21].

D. Objectives and problems

A resource-aware objective (or simply an objective) is a set of loaded runs. The objective S (safety) contains exactly all loaded runs that are safe. Given a target set $T \subseteq S$ and $i \in \mathbb{N}$, the objective R^i_T (bounded reachability) is the set of all safe loaded runs that reach some state from T within the first i steps, which is $\mathsf{R}^i_T = \{{}^d\varrho \in \mathsf{S} \mid \varrho_j \in T \text{ for some } 1 \leq j \leq i+1\}$. The union $\mathsf{R}_T = \bigcup_{i \in \mathbb{N}} \mathsf{R}^i_T$ forms the reachability objective. Finally, the objective B_T (Büchi) contains all safe loaded runs that visit T infinitely often.

The safety objective — never depleting the critical resource — is of primary concern for agents in CMDPs. We reflect this fact in the following definitions. Let us now fix a target set $T \subseteq S$, a state $s \in S$, an initial load d, a strategy σ, and an objective O. We say that σ loaded with d in s

- satisfies O surely, written as $\sigma \, {}^d_s\!\models_{\mathcal{C}} \mathsf{O}$, if and only if ${}^d\varrho \in \mathsf{O}$ holds for every ${}^d\varrho \in \mathsf{Comp}_{\mathcal{C}}(\sigma, s, d)$;
- safely satisfies O with positive probability, written as $\sigma \, {}^d_s\!\models^{>0}_{\mathcal{C}} \mathsf{O}$, if and only if $\sigma \, {}^d_s\!\models_{\mathcal{C}} \mathsf{S}$ and ${}^d_s\mathbb{P}^\sigma_{\mathcal{C}}(\mathsf{O}) > 0$;
- safely satisfies O almost surely, written as $\sigma \, {}^d_s\!\models^{=1}_{\mathcal{C}} \mathsf{O}$, if and only if $\sigma \, {}^d_s\!\models_{\mathcal{C}} \mathsf{S}$ and ${}^d_s\mathbb{P}^\sigma_{\mathcal{C}}(\mathsf{O}) = 1$.

We naturally extend the satisfaction relations to strategies loaded by vectors. Let $\mathbf{x} \in \overline{\mathbb{N}}^S$ be a vector of initial loads. The strategy σ loaded by $\mathbf{x}$ satisfies O, written as $\sigma \, {}^{\mathbf{x}}\!\models_{\mathcal{C}} \mathsf{O}$, if and only if $\sigma \, {}^{\mathbf{x}(s)}_{s}\!\models_{\mathcal{C}} \mathsf{O}$ holds for all $s \in S$ with $\mathbf{x}(s) \neq \infty$. We extend the other two relations analogously to $\sigma \, {}^{\mathbf{x}}\!\models^{>0}_{\mathcal{C}} \mathsf{O}$ and $\sigma \, {}^{\mathbf{x}}\!\models^{=1}_{\mathcal{C}} \mathsf{O}$.

The vector $\mathbf{ml}[\mathsf{O}]_{\mathcal{C}}$ is the component-wise minimal vector for which there exists a strategy π such that $\pi \, {}^{\mathbf{ml}[\mathsf{O}]}\!\models_{\mathcal{C}} \mathsf{O}$. We call π the witness strategy for $\mathbf{ml}[\mathsf{O}]_{\mathcal{C}}$. If $\mathbf{ml}[\mathsf{O}]_{\mathcal{C}}(s) = \infty$, no strategy satisfies O from s even when loaded with *cap*. Vectors $\mathbf{ml}[\mathsf{O}]^{>0}_{\mathcal{C}}$ and $\mathbf{ml}[\mathsf{O}]^{=1}_{\mathcal{C}}$ are defined analogously using $\models^{>0}$ and $\models^{=1}$, respectively.

We consider the following qualitative problems for CMDPs: Safety, positive reachability almost-sure Büchi, and almost-sure reachability which equal to computing $\mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$, $\mathbf{ml}[\mathsf{R}]^{>0}_{\mathcal{C}}$, $\mathbf{ml}[\mathsf{B}]^{=1}$, and $\mathbf{ml}[\mathsf{R}]^{=1}_{\mathcal{C}}$, respectively, and the corresponding witness strategies. The solutions of the latter two problems build on top of the first two.

E. Additional notation and conventions

For given $R' \subseteq S$, we denote by $\mathcal{C}(R')$ the CMDP that uses R' as the set of reloads and otherwise is defined as $\mathcal{C}$. Throughout the paper, we drop the subscripts $\mathcal{C}$ and T in symbols whenever $\mathcal{C}$ or T is known from the context.

Calligraphic font (e.g. $\mathcal{C}$) is used for names of CMDPs, sans serifs (e.g. S) is used for objectives (set of loaded runs), and vectors are written in bold. Action names are letters from the start of the alphabet, while states of CMDPs are usually taken from the latter parts of the alphabet (starting with s). The symbol α is used for both finite and infinite paths, and ϱ is only used for infinite paths (runs). Finally, strategies are always variants of σ or π.

F. Strategies revisited

A strategy σ is memoryless if $\sigma({}^d\alpha) = \sigma({}^h\beta)$ whenever $last(\alpha) = last(\beta)$.

Example 2. *The runs* ϱ *and* ϱ' *from Example 1 are sample runs created by two different memoryless strategies:* σ_{a} *that always picks* a *in* t*, and* σ_{b} *that always picks* b *in* t*, respectively. As* ϱ *is the only* t*-initiated run of* σ_{a}*, we have that* $\sigma_{\mathsf{a}} \, {}^2_t\!\models \mathsf{S}$*. However,* σ_{a} *is not useful if we attempt to eventually*

reach u and we clearly have ${}^{2}_{t}\mathbb{P}^{\sigma_{\mathsf{a}}}(\mathsf{R}_{\{u\}}) = 0$. On the other hand, ϱ' is the witness for the fact that σ_{b} does not even satisfy the safety objective for any initial load. As we have no other choice in t, we can conclude that memoryless strategies are not sufficient in our setting. Consider instead a strategy π that picks b in t whenever the current resource level is at least 10 and picks a (and reloads in s) otherwise. Loaded with 2 in t, π satisfies safety and it guarantees reaching u with a positive probability: in t, we need at least 10 units of resource to return to s in the case we are unlucky and b leads us to v; if we are lucky, b leads us directly to u, witnessing that ${}^{2}_{t}\mathbb{P}^{\pi}(\mathsf{R}_{\{u\}}) > 0$. Moreover, at every revisit of s there is a $\frac{1}{2}$ chance of hitting u during the next attempt, which shows that $\pi \, {}^{2}_{t}\models^{=1} \mathsf{R}_{\{u\}}$.

Remark. While computing the sure satisfaction relation $\models$ on a CMDP follows similar approaches as used for solving a consumption 2-player game [19], the solutions for $\models^{>0}$ and $\models^{=1}$ differ substantially.

The strategy π from Example 2 uses finite memory to track the resource level exactly. The standard results on MDPs with ω-regular objectives [22] show that for the objectives introduced in subsection II-D it is sufficient to consider strategies of the form $S \times \{0, \dots, cap\} \to A$ to achieve optimality (i.e., strategies that decide based on the current state and resource level). A naive way to represent such a strategy is via a table with $|S| \cdot cap$ entries. However, this would be inefficient, since (as we will prove), the optimal strategies are such that in a concrete state s they often select the same action when the resource level falls within some, possibly large interval. To represent such strategies succinctly, we introduce the notion of counter strategies.

III. Strategies with binary counters

Let us fix a CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$. In our setting, strategies need to track resource levels of histories. A non-exhausted resource level is always a number between 0 and cap, which can be represented with a binary-encoded bounded counter. A binary-encoded counter needs $\log_2(cap)$ bits of memory to represent numbers between 0 and cap (the same as integer variables in computers).

We call strategies with such binary-encoded counters finite counter strategies. A finite counter strategy also needs rules that select actions based on the current resource level, and a rule selector that picks the right rule for each state.

Definition 2 (Rule)**.** *A rule φ for $\mathcal{C}$ is a partial function from the set $\{0, \dots, cap\}$ to A. An undefined value for some n is indicated by $\varphi(n) = \bot$.*

We use $dom(\varphi) = \{n \in \{0, \dots, cap\} \mid \varphi(n) \neq \bot\}$ to denote the domain of φ and we call the elements of $dom(\varphi)$ border levels. We use $Rules_{\mathcal{C}}$ for the set of all rules for $\mathcal{C}$.

A rule compactly represents a total function using intervals. Intuitively, the selected action is the same for all values of the resource level in the interval between two border levels. Formally, let l be the current resource level and let $n_1 < n_2 < \cdots < n_k$ be the border levels of φ sorted in the ascending order. Then the selection according to rule φ for l, written as $select(\varphi, l)$, picks the action $\varphi(n_i)$, where n_i is the largest border level such that $n_i \leq l$. In other words, $select(\varphi, l) = \varphi(n_i)$ if the current resource level l is in $[n_i, n_{i+1})$ (putting $n_{k+1} = cap + 1$). We set $select(\varphi, l) = a$ for some globally fixed action $a \in A$ (for completeness) if $l < n_1$. In particular, $select(\varphi, \bot) = a$.

Definition 3 (Rule selector)**.** *A rule selector for $\mathcal{C}$ is a function $\Phi \colon S \to Rules$.*

A binary-encoded counter that tracks the resource levels of paths together with a rule selector Φ encode a strategy σ_Φ. Let ${}^{d}\alpha = {}^{d}s_1 a_1 s_2 \dots s_n$ be a loaded history. We assume that we can access the value of $lastRL({}^{d}\alpha)$ from the counter and we set

$$\sigma_\Phi({}^{d}\alpha) = select(\Phi(s_n), lastRL({}^{d}\alpha)).$$

A strategy σ is a finite counter strategy if there is a rule selector Φ such that $\sigma = \sigma_\Phi$. The rule selector can be imagined as a device that implements σ using a table of size $\mathcal{O}(|S|)$, where the size of each cell corresponds to the number of border levels times $\mathcal{O}(\log cap)$ (the latter representing the number of bits required to encode a level). In particular, if the total number of border levels Φ is polynomial in the size of the MDP, so is the number of bits required to represent Φ (and thus, σ_Φ). This contrasts with the traditional representation of finite-memory strategies via transducers [23], since transducers would require at least $\Theta(cap)$ states to keep track of the current resource level.

Example 3. *Consider the CMDP from Fig. 1. Let φ be a rule with $dom(\varphi) = \{0, 10\}$ such that $\varphi(0) = \mathsf{a}$ and $\varphi(10) = \mathsf{b}$, and let φ' be a rule with $dom(\varphi') = \{0\}$ such that $\varphi(0) = \mathsf{a}$. Finally, let Φ be a rule selector such that $\Phi(s) = \varphi$ and $\Phi(s') = \varphi'$ for all $s \neq s' \in S$. Then, the strategy π informally described in Example 2 can be formally represented by putting $\pi = \sigma_\Phi$. Note that for any $\mathbf{x}$ with $\mathbf{x}(t) \geq 2$, $\mathbf{x}(u) \geq 0$, $\mathbf{x}(v) \geq 5$, $\mathbf{x}(w) \geq 4$, and $\mathbf{x}(s) \geq 0$ we have that $\pi \, {}^{\mathbf{x}}\models^{=1} \mathsf{R}_{\{u\}}$.*

IV. Safety

In this section, we present Algorithm 2 that computes $\mathbf{ml}[\mathsf{S}]$ and the corresponding witness strategy. Such a strategy guarantees that, given a sufficient initial load, the resource will never be depleted regardless the resolution of actions' outcomes. In the remainder of the section we fix an MDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$.

A safe run loaded with d has the following two properties: (i) it never consumes more than cap units of the resource between 2 consecutive visits of reload states, and (ii) it consumes at most d units of the resource (energy) before it reaches the first reload state. To ensure (i), we need to identify a maximal subset $R' \subseteq R$ of reload states for which there is a strategy σ that, starting in some $r \in R'$, can always reach R' again using at most cap resource units. To ensure (ii), we need a strategy that suitably navigates towards R' while not reloading and while using at most d units of the resource.

In summary, for both properties (i) and (ii) we need to find a strategy that can surely reach a set of states (R') without reloading and within a certain limit on consumption (cap and d, respectively). We capture the desired behavior of the strategies by a new objective N (non-reloading reachability).

A. Non-reloading reachability

The problem of non-reloading reachability in CMDPs is similar to the problem of minimum cost reachability on regular MDPs with non-negative costs, which was studied before [24]. In this sub-section, we present a new iterative algorithm for this problem which fits better into our framework and is implemented in our tool. The reachability objective R is defined as a subset of S, and thus relies on resource levels. The following definition of N follows similar ideas as we used for R, but (a) ignores what happens after the first visit of the target set, and (b) it uses the cumulative consumption instead of resource levels to ignore the effect of the reload states.

Given $T \subseteq S$ and $i \in \mathbb{N}$, the objective N^i_T (bounded non-reloading reachability) is the set of all (not necessary safe) loaded runs ${}^d s_1 a_1 s_2 a_2 \dots$ such that for some $1 \leq f \leq i+1$ it holds $s_f \in T$ and $\sum_{j=1}^{f-1} \gamma(s_j, a_j) \leq d$. The union $\mathsf{N}_T = \bigcup_{i \in \mathbb{N}} \mathsf{N}^i_T$ forms the non-reloading reachability objective.

Let us now fix some $T \subseteq S$. In the next few paragraphs, we discuss the solution of sure non-reloading reachability of T: computing the vector $\mathbf{ml}[\mathsf{N}_T]$ and the corresponding witness strategy. The solution is based on backward induction (with respect to number of steps needed to reach T). The key concept here is the value of action a in a state s based on a vector $\mathbf{v} \in \overline{\mathbb{N}}^S$, denoted as $AV(\mathbf{v}, s, a)$ and defined as follows:

$$AV(\mathbf{v}, s, a) = \gamma(s, a) + \max_{t \in Succ(s,a)} \mathbf{v}(t). \tag{2}$$

Intuitively, $AV(\mathbf{v}, s, a)$ is the consumption of a in s plus the worst value of $\mathbf{v}$ among the relevant successors. Now imagine that $\mathbf{v}$ is equal to $\mathbf{ml}[\mathsf{N}^i]$; that is, for each state s it contains the minimal amount of resource needed (without reloading) to reach T in at most i steps. Then, AV for a in s is the minimal amount of resource needed to reach T in $i+1$ steps.

The following functional $\mathcal{F} \colon \overline{\mathbb{N}}^S \to \overline{\mathbb{N}}^S$ is a simple generalization of the standard Bellman functional. We use $\mathcal{F}^i(\mathbf{v})$ for the result of i applications of $\mathcal{F}$ on $\mathbf{v}$,

$$\mathcal{F}(\mathbf{v})(s) = \begin{cases} 0 & s \in T, \\ \min_{a \in A} AV(\mathbf{v}, s, a) & s \notin T. \end{cases} \tag{3}$$

To complete our induction-based computation, we need to find the right initialization vector $\mathbf{x}_T$ for $\mathcal{F}$. As the intuition for action value hints, $\mathbf{x}_T$ should be precisely $\mathbf{ml}[\mathsf{N}^0]$ and thus is defined as $\mathbf{x}_T(s) = 0$ for $s \in T$ and as $\mathbf{x}_T(s) = \infty$ otherwise.

Lemma 1. *It holds that* $\mathbf{ml}[\mathsf{N}^i_T] = \mathcal{F}^i(\mathbf{x}_T)$ *for every* $i \geq 0$.

Proof. We proceed by induction on i. The base case for $i = 0$ is trivial. Now assume that the lemma holds for some $i \geq 0$.

From the definition of N^i we have that a loaded run ${}^d\varrho = {}^d s_1 a_1 s_2 \dots$ satisfies surely N^{i+1} if and only if $s_1 \in T$ or ${}^h\varrho_{2..} \in \mathsf{N}^i$ for $h = d - \gamma(s_1, a_1)$. Therefore, given a state $s \notin T$ and the load $d = \mathbf{ml}[\mathsf{N}^{i+1}](s)$, each witness strategy σ for $\mathbf{ml}[\mathsf{N}^{i+1}]$ must guarantee that if $\sigma({}^d s) = a$ then $d \geq \mathbf{ml}[\mathsf{N}^i](s') + \gamma(s, a)$ for all $s' \in Succ(s, a)$. That is, $d \geq AV(\mathbf{ml}[\mathsf{N}^i], s, a) = AV(\mathcal{F}^i(\mathbf{x}_T), s, a)$.

On the other hand, let a_m be the action with minimal AV for s based on $\mathbf{ml}[\mathsf{N}^i]$. The strategy that plays a_m in the first step and then mimics some witness strategy for $\mathbf{ml}[\mathsf{N}^i]$ surely satisfies N^{i+1} from s loaded by $AV(\mathbf{ml}[\mathsf{N}^i], s, a_m)$. Therefore, $d \leq AV(\mathbf{ml}[\mathsf{N}^i], s, a_m) = AV(\mathcal{F}^i(\mathbf{x}_T), s, a_m)$.

Together, $d = \mathbf{ml}[\mathsf{N}^{i+1}](s) = \min_{a \in A} AV(\mathbf{ml}[\mathsf{N}^i], s, a) = \mathcal{F}(\mathbf{ml}[\mathsf{N}^i])(s)$ and that is by induction hypothesis equal to $\mathcal{F}(\mathcal{F}^i(\mathbf{x}_T))(s) = \mathcal{F}^{i+1}(\mathbf{x}_T)(s)$. □

Theorem 1. *Denote by n the length of the longest simple path in $\mathcal{C}$. Iterating $\mathcal{F}$ on $\mathbf{x}_T$ yields a fixed point in at most n steps and this fixed point equals* $\mathbf{ml}[\mathsf{N}_T]$.

Proof. For the sake of contradiction, suppose that $\mathcal{F}$ does not yield a fixed point after n steps. Then there exists a state s such that $d = \mathcal{F}^{n+1}(\mathbf{x}_T)(s) < \mathcal{F}^n(\mathbf{x}_T)(s)$. Let σ be a witness strategy for $\mathcal{F}^{n+1}(\mathbf{x}_T) = \mathbf{ml}[\mathsf{N}^{n+1}]$. Now let ${}^d\varrho = {}^d s_1 a_1 s_2 \dots$ be a loaded run from $\mathsf{Comp}(\sigma, s, d)$ such that $s_i \notin T$ for all $i \leq n+1$ and $s_{n+2} \in T$, and such that for all $1 \leq k \leq n+1$ it holds $d - c_k = \mathbf{ml}[\mathsf{N}^{n+1-k}](s_{k+1})$ where $c_k = \sum_{j=1}^{k} \gamma(s_j, a_j)$. Such a run must exist, otherwise some $\mathbf{ml}[\mathsf{N}^i]$ can be improved.

As n is the length of the longest simple path in $\mathcal{C}$, we can conclude that there are two indices $f < l \leq n+1$ such that $s_f = s_l = t$. But since $\mathcal{C}$ is decreasing, we have that $c_f < c_l$ and thus $\mathbf{ml}[\mathsf{N}^{n+1-f}](t) = d - c_f > d - c_l = \mathbf{ml}[\mathsf{N}^{n+1-l}](t)$. As $n+1-f > n+1-l$, we reached a contradiction with the fact that $\mathsf{N}^{n+1-f} \supseteq \mathsf{N}^{n+1-l}$.

By Lemma 1 we have that $\mathcal{F}^n(\mathbf{x}_T) = \mathbf{ml}[\mathsf{N}]$. □

Witness strategy for $\mathbf{ml}[\mathsf{N}]$. Any memoryless strategy σ that picks for each history ending with a state s some action a_s such that $AV(\mathbf{ml}[\mathsf{N}], s, a_s) = \mathbf{ml}[\mathsf{N}](s)$ is clearly a witness strategy for $\mathbf{ml}[\mathsf{N}]$, which is, $\sigma^{\,\mathbf{ml}[\mathsf{N}]} \models \mathsf{N}$.

B. Safely reaching reloads from reloads

The objective N is sufficient for the property (ii) with $T = R'$. But we cannot use it off-the-shelf to guarantee the property (i) at most *cap* units of resource are consumed between two consecutive visits of R'. For that, we need to solve the problem of reachability within at least 1 steps (starting in T alone does not count as reaching T here). We define N^i_{+T} in the same way as N_T but we enforce that $f > 1$ and we set $\mathsf{N}_{+T} = \bigcup_{i \in \mathbb{N}} \mathsf{N}^i_{+T}$. To compute $\mathbf{ml}[\mathsf{N}_{+T}]$, we slightly alter $\mathcal{F}$ using the following truncation operator:

$$\llbracket \mathbf{v} \rrbracket_T(s) = \begin{cases} \mathbf{v}(s) & \text{if } s \notin T, \\ 0 & \text{if } s \in T. \end{cases} \tag{4}$$

The new functional $\mathcal{G}$ applied to $\mathbf{v}$ computes the new value in the same way for all states (including states from T), but treats $\mathbf{v}(t)$ as 0 for $t \in T$,

$$\mathcal{G}(\mathbf{v})(s) = \min_{a \in A} AV(\llbracket \mathbf{v} \rrbracket_T, s, a). \tag{5}$$

Let $\boldsymbol{\infty}^S \in \overline{\mathbb{N}}^S$ denote the vector with all components equal to ∞. Clearly, $\llbracket \boldsymbol{\infty}^S \rrbracket_T = \mathbf{x}_T$, and thus it is easy to see that for all $s \notin T$ and $i \in \mathbb{N}$ we have that $\mathcal{G}^i(\boldsymbol{\infty}^S)(s) = \mathcal{F}^i(\mathbf{x}_T)(s)$. Moreover, $\llbracket \mathcal{G}^i(\boldsymbol{\infty}^S) \rrbracket_T = \mathcal{F}^i(\mathbf{x}_T)$. A slight modification of arguments used to prove Lemma 1 and Theorem 1 shows that $\mathcal{G}$ indeed computes $\mathbf{ml}[\mathsf{N}_{+T}]$ and we need at most $n+1$ iterations for the desired fixed point. Algorithm 1 iteratively applies $\mathcal{G}$ on $\boldsymbol{\infty}^S$ until a fixed point is reached.

Algorithm 1: Computing $\mathbf{ml}[\mathsf{N}_{+T}]$.

Input: CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ and $T \subseteq S$
Output: The vector $\mathbf{ml}[\mathsf{N}_{+T}]_{\mathcal{C}}$

1 $\mathbf{v} \leftarrow \boldsymbol{\infty}^S$;
2 **repeat**
3 $\quad \mathbf{v}_{old} \leftarrow \mathbf{v}$;
4 $\quad$ **foreach** $s \in S$ **do**
5 $\quad\quad c \leftarrow \min_{a \in A} AV(\mathbb{\|}\, \mathbf{v}_{old} \,\mathbb{\|}_T, s, a)$;
6 $\quad\quad$ **if** $c < \mathbf{v}(s)$ **then**
7 $\quad\quad\quad \mathbf{v}(s) \leftarrow c$;
8 **until** $\mathbf{v}_{old} = \mathbf{v}$;
9 **return** $\mathbf{v}$

Theorem 2. *Given a CMDP $\mathcal{C}$ and a set of target states T, Algorithm 1 computes the vector $\mathbf{ml}[\mathsf{N}_{+T}]_{\mathcal{C}}$. Moreover, the repeat-loop terminates after at most $|S|$ iterations.*

Proof. Each iteration of the repeat-loop computes an application of $\mathcal{G}$ on the value of $\mathbf{v}$ from line 3 (stored in $\mathbf{v}_{old}$) and stores the resulting values in $\mathbf{v}$ on line 7. Iterating $\mathcal{G}$ on $\boldsymbol{\infty}^S$ yields a fixed point in at most $n+1$ iterations where n is the length of the longest simple path in $\mathcal{C}$. As $n+1 \leq |S|$, the test on line 8 becomes true after at most $|S|$ iterations and $\mathbf{v}$ on line 8 contains the result of $\mathcal{G}^i(\boldsymbol{\infty}^S)$ where i is the actual number of iterations. Thus, the value of $\mathbf{v}$ on line 9 is equal to $\mathbf{ml}[\mathsf{N}_{+T}]_{\mathcal{C}}$ and is computed in at most $|S|$ iterations. □

Now with Algorithm 1 we can compute $\mathbf{ml}[\mathsf{N}_{+R}]_{\mathcal{C}}$ and see which reload states should be avoided by safe runs: the reloads that need more than *cap* units of resource to surely reach R again. We call such reload states unusable in $\mathcal{C}$.

C. Detecting useful reloads and solving the safety problem

Using Algorithm 1, we can identify reload states that are unusable in $\mathcal{C}$. However, it does not automatically mean that the rest of the reload states form the desired set R' for property (i). Consider the CMDP $\mathcal{D}$ in Fig. 2. The only reload state that is unusable is w ($\mathbf{ml}[\mathsf{N}_{+R}](w) = \infty$). But clearly, all runs that avoid w must avoid v and x as well. The property (i) indeed translates to $\mathbf{ml}[\mathsf{N}_{+R'}](r) \leq cap$ for all $r \in R'$; naturally, we want to identify the maximal $R' \subseteq R$ for which this holds. Algorithm 2 finds the desired R' by iteratively removing unusable reloads from the current candidate set R' until there is no unusable reload in R' (lines 3-7).

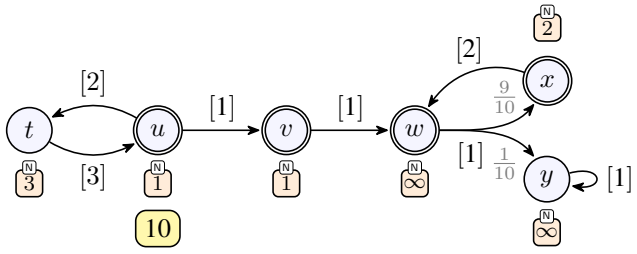


Fig. 2: A CMDP $\mathcal{D}$ with values of $\mathbf{ml}[\mathsf{N}_{+R}]$. For a state s, the value $\mathbf{ml}[\mathsf{N}_{+R}](s)$ is pictured in the orange box below s.

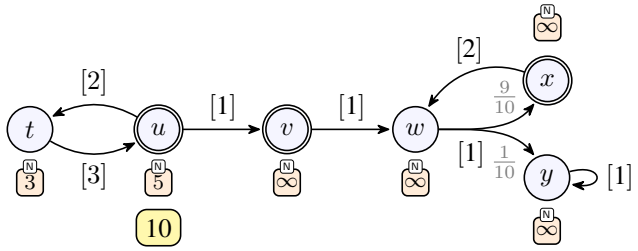


Fig. 3: The CMDP $\mathcal{D}(\{u, v, x\})$ with values of $\mathbf{ml}[\mathsf{N}_{+\{u,v,x\}}]$ in orange boxes. Note the difference in the set of reload states in comparison to Fig. 2.

With the right set R' in hand, we can move on to the property (ii) of safe runs: navigating safely towards reloads in R', which equals to the objective $\mathsf{N}_{R'}$ from Section IV-A. We can reuse Algorithm 1 for it as $\mathbf{ml}[\mathsf{N}] = \mathbb{\|}\, \mathbf{ml}[\mathsf{N}_+] \,\mathbb{\|}$ regardless the target set. Based on properties (i) and (ii), we claim that $\mathbf{ml}[\mathsf{S}] = \mathbb{\|}\, \mathbf{ml}[\mathsf{N}_{+R'}] \,\mathbb{\|}_{R'}$. Indeed, we need at most *cap* units of resource to move between reloads of R', and at most $\mathbb{\|}\, \mathbf{ml}[\mathsf{N}_{+R'}] \,\mathbb{\|}_{R'}(s)$ units of resource to reach R' from s.

Whenever $\mathbf{ml}[\mathsf{S}](s) > cap$ for some state s, the exact value is not important for us; the meaning is still that there is no strategy σ and no initial load $d \leq cap$ such that $\sigma \overset{d}{s} \models \mathsf{S}$. Hence, we can set $\mathbf{ml}[\mathsf{S}](s) = \infty$ in all such cases. To achieve this, we extend the operator $\mathbb{\|} \cdot \mathbb{\|}_T$ into $[\![\cdot]\!]_T^{cap}$ as follows:

$$[\![\, \mathbf{x} \,]\!]_T^{cap}(s) = \begin{cases} \infty & \text{if } \mathbf{x}(s) > cap, \\ \mathbf{x}(s) & \text{if } \mathbf{x}(s) \leq cap \text{ and } s \notin T, \\ 0 & \text{if } \mathbf{x}(s) \leq cap \text{ and } s \in T. \end{cases} \tag{6}$$

Algorithm 2: Computing $\mathbf{ml}[\mathsf{S}]$.

Input: CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$
Output: The vector $\mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$

1 $R' \leftarrow R$;
2 $Unusable \leftarrow \emptyset$;
3 **repeat**
4 $\quad R' \leftarrow R' \setminus Unusable$;
5 $\quad \mathbf{n} \leftarrow \mathbf{ml}[\mathsf{N}_{+R'}]_{\mathcal{C}}$; `/* Algorithm 1 with T = R' */`
6 $\quad Unusable \leftarrow \{r \in R' \mid \mathbf{n}(r) > cap\}$;
7 **until** $Unusable = \emptyset$;
8 **return** $[\![\, \mathbf{n} \,]\!]_{R'}^{cap}$;

Theorem 3. *Algorithm 2 computes the vector $\mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$ in time polynomial with respect to the representation of $\mathcal{C}$.*

Proof. **Complexity.** The algorithm clearly terminates. Computing $\mathbf{ml}[\mathsf{N}_{+R'}]$ on line 5 takes a polynomial number of steps per call (Theorem 2). Since the repeat loop performs at most $|R|$ iterations, the complexity follows.

Correctness. We first prove that upon termination $[\![\, \mathbf{n} \,]\!]_{R'}^{cap}(s) \leq \mathbf{ml}[\mathsf{S}]_{\mathcal{C}}(s)$ for each $s \in S$ whenever the latter value is finite. This is implied by the fact that $\mathbf{ml}[\mathsf{N}_{+R'}] \leq \mathbf{ml}[\mathsf{S}]$ is the invariant of the algorithm. To see that, it suffices to show that at every point of execution, $\mathbf{ml}[\mathsf{S}](t) = \infty$ for each $t \in R \setminus R'$: if this holds, each strategy that satisfies

S must avoid states in $R \setminus R'$ (due to property (i) of safe runs) and thus the first reload on runs compatible with such a strategy must be from R'.

Let R'_i denote the contents of R' after the i-th iteration. We prove, by induction on i, that $\mathbf{ml}[\mathsf{S}](t) = \infty$ for all $t \in R \setminus R'$. For $i = 0$ we have $R = R'_0$, so the statement holds. For $i > 0$, let $t \in R \setminus R'_i$, then it must exist some $j < i$ such that $\mathbf{n}(t) = \mathbf{ml}[\mathsf{N}_{+R'_j}](t) > cap$, hence no strategy can safely reach R'_j from t and by induction hypothesis, the reload states from $R \setminus R'_j$ must be avoided by strategies that satisfy S. Together, as $\mathcal{C}$ is decreasing, there is no strategy σ such that $\sigma \overset{cap}{t} \models \mathsf{S}$ and hence $\mathbf{ml}[\mathsf{S}](t) = \infty$.

Finally, we need to prove that upon termination, $[\![\mathbf{n}]\!]^{cap}_{R'} \geq \mathbf{ml}[\mathsf{S}]$. As $\mathbf{n} = \mathbf{ml}[\mathsf{N}_{+R'}]$ and $\mathbf{n}(r) \leq cap$ for each $r \in R'$, then, for each s with $d = [\![\mathbf{n}]\!]^{cap}_{R'}(s) \leq cap$ there exists a strategy that can reach $R' \subseteq R$ consuming at most d units of resource, and, once in R', σ can always return to R' within cap units of resource. Thus, all runs in $\mathsf{Comp}(\sigma, s, d)$ are safe and d is enough for S in s. □

D. Safe strategies

Definition 4. *Let $s \in S$ be a state and let $0 \leq d \leq cap$ be a resource level. An action a is safe in s with d if (1) $v = AV(\mathbf{ml}[\mathsf{S}], s, a) \leq d$, or if (2) $v \leq cap$ and $s \in R$, or if (3) $\mathbf{ml}[\mathsf{S}](s) > cap$. a is min-safe in s if it is safe in s with $d = \mathbf{ml}[\mathsf{S}](s)$. A strategy σ is safe if it picks safe action in the current state with the current resource level when possible.*

Remarks. By definition, no action is safe in s for all $r < \mathbf{ml}[\mathsf{S}](s) < \infty$ (otherwise $\mathbf{ml}[\mathsf{S}](s)$ is at most r). On the other hand, there is always at least one action that is min-safe for each state s and, in particular, all actions are safe and min-safe in s with $\mathbf{ml}[\mathsf{S}](s) = \infty$.

Lemma 2. *Let σ be a safe strategy. Then $\sigma \overset{\mathbf{ml}[\mathsf{S}]}{} \models \mathsf{S}$.*

Proof. We need to prove that, given a state s with $\mathbf{ml}[\mathsf{S}](s) \leq cap$ and an initial load d such that $\mathbf{ml}[\mathsf{S}](s) \leq d \leq cap$, all runs in $\mathsf{Comp}(\sigma, s, d)$ are safe. To do this, we show that all s-initiated d-loaded paths ${}^d\alpha$ created by σ are safe. By simple induction with respect to the length of the path we prove that $lastRL({}^d\alpha) \geq \mathbf{ml}[\mathsf{S}](t) \neq \bot$ where $t = last(\alpha)$. For ${}^d s$ this clearly holds. Now assume that $lastRL({}^d\alpha) \geq \mathbf{ml}[\mathsf{S}](t) \neq \bot$ for some s-initiated α with $t = last(\alpha)$ and all $d \geq \mathbf{ml}[\mathsf{S}](s)$. Now consider a d'-loaded path ${}^{d'}s'as \odot \alpha$ created by σ; we have $lastRL({}^{d'}s'as) = d' - \gamma(s', a) \geq \mathbf{ml}[\mathsf{S}](s)$ by definition of action value and safe actions, and thus by the induction hypothesis we have that $lastRL({}^{d'}s'as \odot \alpha) \geq \mathbf{ml}[\mathsf{S}](t)$. □

Theorem 4. *In each consumption MDP $\mathcal{C}$ there is a memoryless strategy σ such that $\sigma \overset{\mathbf{ml}[\mathsf{S}]_\mathcal{C}}{} \models_\mathcal{C} \mathsf{S}$. Moreover, σ can be computed in time polynomial w.r.t the representation of $\mathcal{C}$.*

Proof. Using Lemma 2, the existence of a memoryless strategy follows from the fact that a strategy that fixes one min-safe action in each state is safe. The complexity follows from Theorem 3. □

Example 4. *Figure 4 shows again the CMDP from Fig. 1 and includes also values computed by Algorithms 1 and 2. Algorithm 2 stores the values of $\mathbf{ml}[\mathsf{N}]$ into $\mathbf{n}$ and, because no value is ∞, returns just $[\![\mathbf{n}]\!]^{cap}_R$. The strategy σ_a from Example 2 is a witness strategy for $\mathbf{ml}[\mathsf{S}]$. As $\mathbf{ml}[\mathsf{S}](t) = 2$, no strategy would be safe from t with initial load 1.*

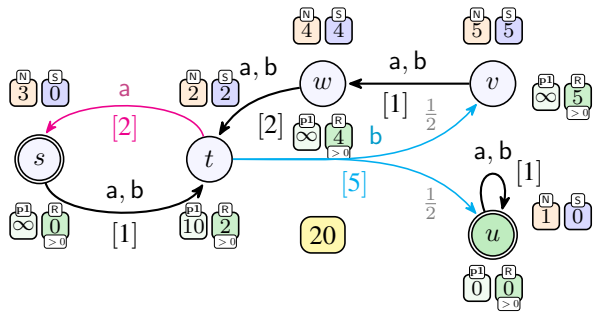


Fig. 4: The CMDP from Fig. 1 with vectors for Examples 4 and 5. The values of $\mathbf{ml}[\mathsf{N}_R]$ and $\mathbf{ml}[\mathsf{S}]$ (referenced in Example 4) are indicated by the orange (left) and blue (right) boxes above states, respectively. Values of $\mathbf{p}1$ (left) and $\mathbf{ml}[\mathsf{R}]^{>0}$ (right) that are referenced in Example 5 are indicated by the green boxes below states.

V. Positive reachability

In this section, we present the solution of the problem called positive reachability. We focus on strategies that, given a set $T \subseteq S$ of target states, safely satisfy $\mathsf{R}_T \subseteq \mathsf{S}$, i.e., the reachability objective, with a positive probability. The main contribution of this section is Algorithm 3 that computes $\mathbf{ml}[\mathsf{R}]^{>0}$ and the corresponding witness strategy. As before, for the rest of this section we fix a CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ and also a set $T \subseteq S$.

Let $s \in S \setminus T$ be a state, let d be an initial load and let σ be a strategy such that $\sigma \overset{d}{s} \models^{>0} \mathsf{R}_T$. Intuitively, as $\sigma \overset{d}{s} \models^{>0} \mathsf{R}$ implies $\sigma \overset{d}{s} \models \mathsf{S}$, the strategy is limited to safe actions. For the reachability part, σ must start with an action $a = \sigma({}^d s)$ such that for at least successor s' of a in s it holds that $\sigma \overset{d'}{s'} \models^{>0} \mathsf{R}$ with $d' = d - \gamma(s, a)$. It must then continue in a similar fashion from s' until either T is reached (and σ produces the desired run from R_T) or until there is no such action.

To formalize the intuition, we define two auxiliary functions. Let us fix a state s, and action a, a successor $s' \in Succ(s, a)$, and a vector $\mathbf{x} \in \overline{\mathbb{N}}^S$. We define the hope value of s' for a in s based on $\mathbf{x}$, denoted by $HV(\mathbf{x}, s, a, s')$, and the safe value of a in s based on $\mathbf{x}$, denoted by $SV(\mathbf{x}, s, a)$, as follows: $HV(\mathbf{x}, s, a, s') = \max_{t \in Succ(s,a) t \neq s'} \{\mathbf{x}(s'), \mathbf{ml}[\mathsf{S}](t)\}$, $SV(\mathbf{x}, s, a) = \gamma(s, a) + \min_{s' \in Succ(s,a)} HV(\mathbf{x}, s, a, s')$.

The hope value of s' for a in s represents the lowest level of resource that the agent needs to have after playing a in order to (i) have at least $\mathbf{x}(s')$ units of resource if the outcome of a is s', and (ii) to survive otherwise. On the other hand, the safe value represents the consumption of a in s plus the least hope value among the relevant successors.

We again use functionals to iteratively compute $\mathbf{ml}[\mathsf{R}^i]^{>0}$, with a fixed point equal to $\mathbf{ml}[\mathsf{R}]^{>0}$. The main operator $\mathcal{B}$ just applies $[\![\cdot]\!]^{cap}_R$ on the result of the auxiliary functional $\mathcal{A}$.

Algorithm 3: Computing $\mathbf{ml}[\mathsf{R}_T]^{>0}$ and a corresponding witness rule selector.

Input: CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ and $T \subseteq S$
Output: The vector $\mathbf{ml}[\mathsf{R}_T]^{>0}_{\mathcal{C}}$, rule selector Φ

1 compute $\mathbf{ml}[\mathsf{S}]$; /* Algorithm 2 */
2 **foreach** $s \in S$ **do**
3 $\quad \Phi(s)(\mathbf{ml}[\mathsf{S}](s)) \leftarrow$ arbitrary min-safe action of s
4 $\mathbf{p} \leftarrow \{\infty\}^S$;
5 **foreach** $t \in T$ **do** $\mathbf{p}(t) \leftarrow \mathbf{ml}[\mathsf{S}](t)$;
6 **repeat**
7 $\quad \mathbf{p}_{old} \leftarrow \mathbf{p}$;
8 $\quad$ **foreach** $s \in S \setminus T$ **do**
9 $\quad\quad \mathbf{a}(s) \leftarrow \arg\min_{a \in A} SV(\mathbf{p}_{old}, s, a)$;
10 $\quad\quad \mathbf{p}(s) \leftarrow \min_{a \in A} SV(\mathbf{p}_{old}, s, a)$;
11 $\quad \mathbf{p} \leftarrow [\![\mathbf{p}]\!]^{cap}_R$;
12 $\quad$ **foreach** $s \in S \setminus T$ **do**
13 $\quad\quad$ **if** $\mathbf{p}(s) < \mathbf{p}_{old}(s)$ **then**
14 $\quad\quad\quad \Phi(s)(\mathbf{p}(s)) \leftarrow \mathbf{a}(s)$;
15 **until** $\mathbf{p}_{old} = \mathbf{p}$;
16 **return** $\mathbf{p}, \Phi$;

The application of $[\![\cdot]\!]^{cap}_R$ ensures that whenever the result is higher than cap, it set to ∞, and that in reload states the value is either 0 or ∞ – in line what is expected from $\mathbf{ml}[\mathsf{R}]^{>0}$.

$$\mathcal{A}(\mathbf{x})(s) = \begin{cases} \mathbf{ml}[\mathsf{S}](s) & \text{if } s \in T, \\ \min\limits_{a \in A} SV(\mathbf{x}, s, a) & \text{otherwise;} \end{cases} \tag{7}$$

$$\mathcal{B}(\mathbf{x}) = [\![\mathcal{A}(\mathbf{x})]\!]^{cap}_R. \tag{8}$$

By $\mathbf{y}_T$ we denote a vector such that

$$\mathbf{y}_T(s) = \begin{cases} \mathbf{ml}[\mathsf{S}](s) & \text{if } s \in T, \\ \infty & \text{if } s \notin T. \end{cases} \tag{9}$$

The following two lemmata relate $\mathcal{B}$ to $\mathbf{ml}[\mathsf{R}]^{>0}$ and show that $\mathcal{B}$ applied iteratively on $\mathbf{y}_T$ reaches fixed point in a number of iterations that is polynomial with respect to the representation of $\mathcal{C}$. Their proofs are provided in the Appendix.

Lemma 3. *Consider the iteration of $\mathcal{B}$ on the initial vector $\mathbf{y}_T$. Then for each $i \geq 0$ it holds that $\mathcal{B}^i(\mathbf{y}_T) = \mathbf{ml}[\mathsf{R}^i]^{>0}$.*

Lemma 4. *Let $K = |R| + (|R| + 1) \cdot (|S| - |R| + 1)$. Then $\mathcal{B}^K(\mathbf{y}_T) = \mathbf{ml}[\mathsf{R}]^{>0}$.*

Algorithm 3 computes $\mathbf{ml}[\mathsf{R}]^{>0}$ and a corresponding witness rule selector Φ. On lines 4 and 5 $\mathbf{p}$ is initialized to $\mathbf{y}_T$. The repeat-loop on Lines 6 to 15 iterates $\mathcal{B}$ on $\mathbf{p}$ (and $\mathbf{p}_{old}$) and builds the witness selector gradually. In particular, the Line 10 stores the application of $\mathcal{A}$ on $\mathbf{p}_{old}$ in $\mathbf{p}$, the Line 11 performs $\mathcal{B}$, and the condition on Line 15 checks for the fixed point. Finally, lines 9 and 12-14 update Φ accordingly. The correctness and complexity of the algorithm are stated formally in theorems 5 and 6.

Example 5. *Consider again Fig. 4 which shows the values of* $\mathbf{p1}$*: the vector $\mathbf{p}$ after one iteration of the repeat loop of Algorithm 3. In this iteration, the algorithm set Φ to play* b *in t with resource level 10 or more. The final values of $\mathbf{p} = \mathbf{ml}[\mathsf{R}]^{>0}$ computed by Algorithm 3 are equal to $\mathbf{ml}[\mathsf{S}]$ for this example (as we can safely reach u from all reloads). In the iteration, where $\mathbf{p}(t) = 2$ for the first time, the selector is updated to play* a *in t with resource level between 2 and 10 (excluded).*

Theorem 5. *Algorithm 3 always terminates after a number of steps that is polynomial with respect to the representation of $\mathcal{C}$, and upon termination, $\mathbf{p} = \mathbf{ml}[\mathsf{R}]^{>0}$.*

Proof. The complexity part follows from Lemma 4 and the fact that each iteration takes only linear number of steps. The correctness part is an immediate corollary of Lemma 4 and the fact that Algorithm 3 iterates $\mathcal{B}$ on $\mathbf{y}_T$ until a fixed point. □

Theorem 6. *Upon termination of Algorithm 3, the computed rule selector Φ encodes a strategy σ_Φ, $\sigma_\Phi \models^{\mathbf{v}} \mathsf{R}^{>0}$ for $\mathbf{v} = \mathbf{ml}[\mathsf{R}]^{>0}$. As a consequence, a polynomial-size finite counter strategy for the positive reachability problem can be computed in time polynomial with respect to representation of $\mathcal{C}$.*

Proof. The complexity follows from Theorem 5. Indeed, since the algorithm has polynomial complexity, also the size of Φ is polynomial. The correctness proof is based on the following invariant of the main repeat-loop. The vector $\mathbf{p}$ and the finite counter strategy $\pi = \sigma_\Phi$ have these properties:

(a) It holds that $\mathbf{p} \geq \mathbf{ml}[\mathsf{S}]$.
(b) Strategy π is safe.
(c) For each $s \in S$ with d such that $\mathbf{p}(s) \leq d \leq cap$, there is a finite π-compatible path ${}^d\alpha = {}^d s_1 a_1 s_2 \dots s_n$ with $s_1 = s$ and $s_n \in T$ such that $RL({}^d\alpha) = r_1, r_2, \dots, r_n$ never drops below $\mathbf{p}$, which is $r_i \geq \mathbf{p}(s_i)$ for all $1 \leq i \leq n$.

The theorem then follows from (b) and (c) of this invariant and from Theorem 5.

Clearly, all parts of the invariants hold after the initialization on Lines 2 to 5. The first item of the invariant follows from the definition of SV and HV. In particular, if $\mathbf{p}_{old} \geq \mathbf{ml}[\mathsf{S}]$, then $SV(\mathbf{p}_{old}, s, a) \geq AV(\mathbf{ml}[\mathsf{S}], s, a) \geq \mathbf{ml}[\mathsf{S}](s)$ for all s and a. The part (b) follows from (a), as the action assigned to Φ on Line 14 is safe for s with $\mathbf{p}(s)$ units of resource (again, due to $\mathbf{p}(s) = SV(\mathbf{p}_{old}, s, a) \geq AV(\mathbf{ml}[\mathsf{S}], s, a)$); hence, only actions that are safe for the corresponding state and resource level are assigned to Φ. By Lemma 2, π is safe.

The proof of (c) is more involved. Assume that an iteration of the main repeat loop was performed. Denote by π_{old} the strategy encoded by $\mathbf{p}$ and Φ from the previous iteration. Let s be any state such that $\mathbf{p}(s) \leq cap$. If $\mathbf{p}(s) = \mathbf{p}_{old}(s)$, then (c) follows directly from the induction hypothesis: for each state q, $\Phi(q)$ was only redefined for values smaller then $\mathbf{p}_{old}(q)$ and thus the history witnessing (c) for π_{old} is also π-compatible.

The case where $\mathbf{p}(s) < \mathbf{p}_{old}(s)$ is treated similarly. We denote by a the action $\mathbf{a}(s)$ selected on Line 9 and assigned to $\Phi(s)$ for $\mathbf{p}(s)$ on line 14. By definition of SV, there must be $t \in Succ(s, a)$ such that $HV(\mathbf{p}_{old}, s, a, t) + \gamma(s, a) = SV(\mathbf{p}_{old}, s, a)$ (which is equal to $\mathbf{p}(s)$ before the truncation on Line 11). In particular, it holds that $l = lastRL(\mathbf{p}^{(s)}sat) \geq \mathbf{p}_{old}(t)$ (even after the truncation). Then, by the induction

Algorithm 4: Computing $\mathbf{ml}[\mathsf{B}_T]^{=1}$ and a corresponding witness rule selector.

Input: CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ and $T \subseteq S$
Output: The vector $\mathbf{ml}[\mathsf{B}_T]^{=1}_{\mathcal{C}}$, rule selector Φ

$R' \leftarrow R$; $Unusable \leftarrow \emptyset$;
repeat
  $R' \leftarrow R' \setminus Unusable$;
  /* The next 2 lines use Algorithm 3 on $\mathcal{C}(R')$ and T. */
  $\mathbf{b} \leftarrow \mathbf{ml}[\mathsf{R}_T]^{>0}_{\mathcal{C}(R')}$;
  $\Phi \leftarrow$ witness selector for $\mathbf{ml}[\mathsf{R}_T]^{>0}_{\mathcal{C}(R')}$;
  $Unusable \leftarrow \{r \in R' \mid \mathbf{b}(r) > cap\}$;
until $Unusable = \emptyset$;
return $\mathbf{b}, \Phi$

hypothesis, there is a t-initiated finite path β witnessing (c) for π_{old}. Then, the loaded history ${}^{\mathbf{p}(s)}\alpha$ with $\alpha = sat \odot \beta$ is (i) compatible with π and, moreover, (ii) we have that $RL({}^{\mathbf{p}_{old}(s)}\alpha)$ never drops below $\mathbf{p}$. Indeed, (i) $\Phi(s)(\mathbf{p}(s)) = a$ (see Line 14), and (ii) Φ was only redefined for values lower than $\mathbf{p}_{old}$ and thus π mimics π_{old} from t onward. For the initial load $\mathbf{p}(s) < d' \leq cap$ the same arguments apply. This finishes the proof of the invariant and also the proof of Theorem 6. □

VI. Büchi: visiting targets repeatedly

This section solves the almost-sure Büchi problem. As before, for the rest of this section we fix a CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ and a set $T \subseteq S$.

The solution builds on the positive reachability problem similarly to how the safety problem builds on the nonreloading reachability problem. In particular, we identify a largest set $R' \subseteq R$ such that from each $r \in R'$ we can safely reach R' again (in at least one step) while restricting ourselves only to safe strategies that (i) avoid $R \setminus R'$ and (ii) guarantee positive reachability of T in $\mathcal{C}(R')$ from all $r \in R'$.

Intuitively, at each visit of R', such a strategy can attempt to reach T. With an infinite number of attempts, we reach T infinitely often with probability 1 (almost surely). Formally, we show that for a suitable R' we have that $\mathbf{ml}[\mathsf{B}_T]^{=1}_{\mathcal{C}} = \mathbf{ml}[\mathsf{R}_T]^{>0}_{\mathcal{C}(R')}$ (where $\mathcal{C}(R')$ denotes the CMDP defined as $\mathcal{C}$ with the exception that R' is the set of reload states).

Algorithm 4 identifies the suitable set R' using Algorithm 3 in a similar fashion as Algorithm 2 handled safety using Algorithm 1. In each iteration, we declare as non-reload states all states of R from which positive reachability of T within $\mathcal{C}(R')$ cannot be guaranteed. This is repeated until we reach a fixed point. The number of iterations is bounded by $|R|$.

Theorem 7. *Upon termination of Algorithm 4, for the strategy σ_Φ encoded by Φ it holds $\sigma_\Phi \models^{\mathbf{b}}_{\mathcal{C}} \mathsf{B}$. Moreover, $\mathbf{b} = \mathbf{ml}[\mathsf{B}]^{=1}_{\mathcal{C}}$. As a consequence, a polynomial-size finite counter strategy for the almost-sure Büchi problem can be computed in time polynomial with respect to the representation of $\mathcal{C}$.*

Proof. The complexity part follows from the fact that the number of iterations of the repeat-loop is bounded by $|R|$ and from theorems 5 and 6.

For the correctness part, we first prove that $\sigma_\Phi \models^{\mathbf{b}}_{\mathcal{C}(R')} \mathsf{B}$. Then we argue that the same holds also for $\mathcal{C}$. Finally, we show that $\mathbf{b} \leq \mathbf{ml}[\mathsf{B}]^{=1}_{\mathcal{C}}$; the converse follows from $\sigma_\Phi \models^{\mathbf{b}}_{\mathcal{C}} \mathsf{B}$.

Strategy σ_Φ has finite memory, also $\sigma_\Phi \models^{\mathbf{b}}_{\mathcal{C}(R')} \mathsf{R}$, and upon termination, $\mathbf{b}(r) = 0$ for all $r \in R'$. Therefore, there is $\theta > 0$ such that upon every visit of some state $r \in R'$ we have that ${}^{0}_{r}\mathbb{P}^{\sigma_\Phi}_{\mathcal{C}(R')}(\mathsf{R}) \geq \theta$.

As $\mathcal{C}(R')$ is decreasing, every safe infinite run created by σ_Φ in $\mathcal{C}(R')$ must visit R' infinitely many times. Hence, with probability 1 we reach T at least once. The argument can then be repeated from the first point of visit of T to show that with probability 1 we visit T at least twice, three times, etc. ad infinitum. By the monotonicity of probability, we get ${}^{d}_{s}\mathbb{P}^{\sigma_\Phi}_{\mathcal{C}(R')}(\mathsf{B}) = 1$ for all $s : d = \mathbf{b}(s) \leq cap$ and $\sigma_\Phi \models^{\mathbf{b}}_{\mathcal{C}(R')} \mathsf{B}$.

Let s be a state such that $\mathbf{b}(s) \leq cap$. Clearly, all s-initiated runs loaded by $d \geq \mathbf{b}(s)$ that are compatible with σ_Φ in $\mathcal{C}(R')$ avoid $R \setminus R'$. Therefore, $\mathsf{Comp}_{\mathcal{C}}(\sigma_\Phi, s, d) = \mathsf{Comp}_{\mathcal{C}(R')}(\sigma_\Phi, s, d)$ and we also get $\sigma_\Phi \models^{d}_{s} \mathsf{B}$.

It remains to show that $\mathbf{b} \leq \mathbf{ml}[\mathsf{B}]^{=1}_{\mathcal{C}}$. Assume for the sake of contradiction that there is a state $s \in S$ and a strategy σ such that $\sigma \models^{d}_{s} \mathsf{B}$ for some $d < \mathbf{b}(s) = \mathbf{ml}[\mathsf{R}]^{>0}_{\mathcal{C}(R')}(s)$. Then there must be at least one ${}^{d}\alpha$ created by σ such that ${}^{d}\alpha$ visits $r \in R \setminus R'$ before reaching T (otherwise $d \geq \mathbf{b}(s)$). Then either (a) $\mathbf{ml}[\mathsf{R}]^{>0}_{\mathcal{C}}(r) = \infty$, in which case any σ-compatible extension of ${}^{d}\alpha$ avoids T; or (b) since $\mathbf{ml}[\mathsf{R}]^{>0}_{\mathcal{C}(R')}(r) > cap$, there must be an extension of α that visits, between the visit of r and T, another $r' \in R \setminus R'$ such that $r' \neq r$. We can then repeat the argument, eventually reaching the case (a) or running out of the resource, a contradiction with $\sigma \models^{d}_{s} \mathsf{S}$. □

VII. Almost-sure reachability

In this section, we solve the almost-sure reachability problem by computing the vector $\mathbf{ml}[\mathsf{R}_T]^{=1}$ and the corresponding witness strategy for a given set of target states $T \subseteq S$.

A. Reduction to Büchi

In the absence of the resource constraints, reachability can be viewed as a special case of Büchi: we can simply modify the MDP so that playing any action in some target state $t \in T$ results into looping in t, thus replacing reachability with an equivalent Büchi condition. In consumption MDPs, the transformation is slightly more involved, due to the need to "survive" after reaching T. Hence, for every CMDP $\mathcal{C}$ and a target set T we define a new CMDP $\mathcal{B}(\mathcal{C}, T)$ so that solving $\mathcal{B}(\mathcal{C}, T)$ w.r.t. the Büchi objective entails solving $\mathcal{C}$ w.r.t. the reachability objective. Formally, for $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ we have $\mathcal{B}(\mathcal{C}, T) = (S', A, \Delta', \gamma', R', cap)$, where the differing components are defined as follows:

- $S' = S \cup \{sink\}$, where $sink \notin S$ is a new sink state, i.e. $\Delta'(sink, a, sink) = 1$ for each $a \in A$;
- we have $R' = R \cup \{sink\}$;
- for each $t \in T$ and $a \in A$ we have $\Delta'(t, a, sink) = 1$ and $\Delta'(t, a, s) = 0$ for all $s \in S$;
- for each $t \in T$ and $a \in A$ we have $\gamma'(t, a) = \mathbf{ml}[\mathsf{S}]_{\mathcal{C}}(t)$;
- we have $\gamma'(sink, a) = 1$ for each $a \in A$; and
- we have $\Delta'(s, a, t) = \Delta(s, a, t)$ and $\gamma'(s, a) = \gamma(s, a)$ for every $s \in S \setminus T$, every $a \in A$, and every $t \in S$.

We can easily prove the following:

Lemma 5. *For every $s \in S$ it holds that $\mathbf{ml}[\mathsf{R}_T]^{=1}_{\mathcal{C}}(s) = \mathbf{ml}[\mathsf{B}_{\{sink\}}]^{=1}_{\mathcal{B}(\mathcal{C},T)}(s)$. Moreover, from a witness strategy for $\mathbf{ml}[\mathsf{R}_T]^{=1}_{\mathcal{C}}$ we can extract, in time polynomial with respect to the representation of $\mathcal{C}$, the witness strategy for $\mathbf{ml}[\mathsf{B}_{\{sink\}}]^{=1}_{\mathcal{B}(\mathcal{C},T)}$ and vice versa.*

Proof. Let σ be a witness strategy for $\mathbf{ml}[\mathsf{B}_{\{sink\}}]^{=1}_{\mathcal{B}(\mathcal{C},T)}$ in $\mathcal{B}(\mathcal{C},T)$. Consider a strategy π in $\mathcal{C}$ which, starting in some state s, mimics σ until some $t \in T$ is reached, and then switches to mimicking an arbitrary safe strategy. Since σ reaches $sink$ and thus also T almost-surely, so does π. Moreover, since σ is safe, upon reaching a $t \in T$ the current resource level is at least $\mathbf{ml}[\mathsf{S}]_{\mathcal{C}}(t)$, since consuming this amount is enforced in the next step. This is sufficient for π to prevent resource exhaustion after switching to a safe strategy.

It follows that $\mathbf{ml}[\mathsf{B}_{\{sink\}}]^{=1}_{\mathcal{B}(\mathcal{C},T)}(s) \geq \mathbf{ml}[\mathsf{R}_T]^{=1}_{\mathcal{C}}(s)$ for all $s \in S$. The converse inequality can be proved similarly, by defining a straightforward conversion of a witness strategy for $\mathbf{ml}[\mathsf{R}_T]^{=1}_{\mathcal{C}}$ into a witness strategy for $\mathbf{ml}[\mathsf{B}_{\{sink\}}]^{=1}_{\mathcal{B}(\mathcal{C},T)}(s)$. The conversion can be clearly performed in polynomial time, with the help of Algorithm 2. □

Hence, we can solve almost-sure reachability for $\mathcal{C}$ by constructing $\mathcal{B}(\mathcal{C},T)$ and solving the latter for almost-sure Büchi via Algorithm 4. The construction of $\mathcal{B}(\mathcal{C},T)$ can be clearly performed in time polynomial in the representation of $\mathcal{C}$ (using Algorithm 2 to compute $\mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$), hence also almost-sure reachability can be solved in polynomial time.

B. Almost-sure reachability without model modification

In practice, building $\mathcal{B}(\mathcal{C},T)$ and translating the synthesized strategy back to $\mathcal{C}$ is inconvenient. Hence, we also present an algorithm to solve almost-sure reachability directly on $\mathcal{C}$. The algorithm consists of a minor modification of the already presented algorithms.

To argue the correctness of the algorithm, we need a slight generalization of the MDP modification. We call a vector $\mathbf{v} \in \overline{\mathbb{N}}^S$ a sink vector for $\mathcal{C}$ if and only if $0 \leq \mathbf{v}(s) \leq \mathbf{ml}[\mathsf{S}]^{=1}_{\mathcal{C}}(s)$ or $\mathbf{v}(s) = \infty$ for all $s \in S$. By $F(\mathbf{v})$ we denote the set $\{s \in S \mid \mathbf{v}(s) < \infty\}$ of states with finite value of $\mathbf{v}$ and we call each member of this set a sink entry. We say that $\mathbf{v}$ is a sink vector for T if $F(\mathbf{v}) = T$. Given a CMDP $\mathcal{C}$, target set T, and sink vector $\mathbf{v}$ for T, we define a new CMDP $\mathcal{B}(\mathcal{C},T,\mathbf{v})$ in exactly the same way as $\mathcal{B}(\mathcal{C},T)$, except for the fourth point: for every $t \in T$ we put $\gamma'(t,a) = \mathbf{v}(t)$ for all $a \in A$. Note that $\mathcal{B}(\mathcal{C},T) = \mathcal{B}(\mathcal{C},T,\mathbf{ml}[\mathsf{S}]_{\mathcal{C}})$.

Given a CMDP $\mathcal{C}$, subsets of states T, X of $\mathcal{C}$, and a sink vector $\mathbf{v}$ for T, Algorithm 5 computes the S-components of the vector $\mathbf{ml}[\mathsf{N}_{+X'}]_{\mathcal{B}(\mathcal{C},T,\mathbf{v})}$, where $X' = X \cup \{sink\}$. To see this, denote by $\mathbf{x}_i$ the contents of the variable $\mathbf{x}$ in the i-th iteration of Algorithm 5 on the input $\mathcal{C}$, T, X, $\mathbf{v}$; and by $\mathbf{v}_i$ the contents of variable $\mathbf{v}$ in the i-th iteration of an execution of Algorithm 1 on CMDP $\mathcal{B}(\mathcal{C},T,\mathbf{v})$ with target set $X \cup \{sink\}$. Induction shows that $\mathbf{v}_i(s) = \mathbf{x}_i(s)$ for all $s \in S$ and all i.

Then, the vector $\mathbf{ml}[\mathsf{S}]_{\mathcal{B}(\mathcal{C},T,\mathbf{v})}$ can be computed using a slight modification of Algorithm 2: on Line 5 use

Algorithm 5: Modified safe sure reachability.

Input: CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$, sets of states $T, X \subseteq S$, a sink vector $\mathbf{v} \in \overline{\mathbb{N}}^S$ for T
Output: $\mathbf{ml}[\mathsf{N}_{+X\cup\{sink\}}]_{\mathcal{B}(\mathcal{C},T,\mathbf{v})}$ projected to S

1 $\mathbf{x} \leftarrow \infty^S$;
2 **repeat**
3 $\mathbf{x}_{old} \leftarrow \mathbf{x}$;
4 **foreach** $s \in S$ **do**
5 **if** $s \in F(\mathbf{v})$ **then**
6 $\mathbf{x}(s) \leftarrow \mathbf{v}(s)$;
7 **else**
8 $c \leftarrow \min_{a \in A} AV(\llbracket \mathbf{x}_{old} \rrbracket_X, s, a)$;
9 **if** $c < \mathbf{x}(s)$ **then**
10 $\mathbf{x}(s) \leftarrow c$;
11 **until** $\mathbf{x}_{old} = \mathbf{x}$;
12 **return** $\mathbf{x}$

$\mathbf{ml}[\mathsf{N}_{+R'\cup\{sink\}}]_{\mathcal{B}(\mathcal{C},T,\mathbf{v})}$ (projected to S) computed by Algorithm 5 instead of $\mathbf{ml}[\mathsf{N}_{+R'}]$ computed by Algorithm 1. Then, run the modified algorithm on $\mathcal{C}$. The correctness can be argued similarly as for Algorithm 5: let R'_i be the contents of R' in the i-th iteration of Algorithm 2 on $\mathcal{B}(\mathcal{C},T,\mathbf{v})$; and let $\tilde{R}'_i$ be the contents of R' in the i-th iteration of the modified algorithm executed on $\mathcal{C}$. Clearly, $sink \in \tilde{R}'_i$ for all i. An induction on i shows that for all i we have $R' = \tilde{R}' \setminus \{sink\}$, so both algorithms terminate in the same iteration. The correctness follows from Algorithm 2.

Now we can proceed to solve almost-sure reachability. Algorithm 6 combines (slightly modified) algorithms 3 and 4 to mimic the solving of Büchi objective in $\mathcal{B}(\mathcal{C},T)$ with a single Büchi accepting state $\{sink\}$.

Lines 7-21 correspond to the computation of Algorithm 3 on $\mathcal{B}(\mathcal{C},T)(R' \cup \{sink\})$. To see this, note that $\mathcal{B}(\mathcal{C},T)(R' \cup \{sink\}) = \mathcal{B}(\mathcal{C}(R'),T,\mathbf{v})$ for $\mathbf{v} = \mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$. Hence, line 1 in Algorithm 3 is replaced by line 11 in Algorithm 6. Also, on lines 15 and 16, we use versions of HV and SV that use survival values $\mathbf{s}$: $HV[\mathbf{s}](\mathbf{x},s,a,s') = \max_{t \in Succ(s,a) t \neq s'}\{\mathbf{x}(s'),\mathbf{s}(t)\}$, $SV[\mathbf{s}](\mathbf{x},s,a) = \gamma(s,a) + \min_{s' \in Succ(s,a)} HV[\mathbf{s}](\mathbf{x},s,a,s')$. Note that $HV = HV[\mathbf{ml}[\mathsf{S}]]$ and $SV = SV[\mathbf{ml}[\mathsf{S}]]$. These generalized operators are used because Algorithm 6 works on $\mathcal{C}$, but lines 15 and 16 should emulate the computation of lines 9 and 10 on $\mathcal{B}(\mathcal{C},T)(R' \cup \{sink\})$, so the vector $\mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$ in the definition of the hope value HV has to be substituted for $\mathbf{ml}[\mathsf{S}]_{\mathcal{B}(\mathcal{C},T)(R'\cup\{sink\})} = \mathbf{ml}[\mathsf{S}]_{\mathcal{B}(\mathcal{C}(R'),T,\mathbf{v})}$ where $\mathbf{v} = \mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$.

Hence, the respective lines indeed emulate the computation of Algorithm 3 on $\mathcal{B}(\mathcal{C},T)(R' \cup \{sink\})$. It remains to show that the whole repeat loop emulates the computation of Algorithm 4 on $\mathcal{B}(\mathcal{C},T)$. But this follows immediately from the fact that in the latter computation, $sink$ always stays in R'.

VIII. Improving Expected Reachability Time

The number of steps that a strategy needs on average to reach the target set T (expected reachability time (ERT)) is

Algorithm 6: Computing $\mathbf{ml}[\mathsf{R}_T]^{=1}$ and a corresponding witness rule selector.

Input: CMDP $\mathcal{C} = (S, A, \Delta, \gamma, R, cap)$ and $T \subseteq S$
Output: The vector $\mathbf{ml}[\mathsf{R}_T]^{=1}_{\mathcal{C}}$, rule selector Φ

1 $\mathbf{o} \leftarrow \mathbf{ml}[\mathsf{S}]_{\mathcal{C}}$; /* Algorithm 2 */
2 $\mathbf{v} \leftarrow \{\infty\}^S$;
3 **foreach** $t \in T$ **do** $\mathbf{v}(t) \leftarrow \mathbf{o}(t)$;
4 $R' \leftarrow R$; $\mathit{Unusable} \leftarrow \emptyset$;
5 **repeat**
6 $\quad R' \leftarrow R' \setminus \mathit{Unusable}$;
7 $\quad \Phi \leftarrow$ an empty selector;
8 $\quad$ **foreach** $s \in S$ **do**
9 $\quad\quad \Phi(s)(\mathbf{o}(s)) \leftarrow$ arbitrary min-safe action of s
10 $\quad \mathbf{p} \leftarrow \mathbf{v}$;
11 $\quad \mathbf{s} \leftarrow \mathbf{ml}[\mathsf{S}]_{\mathcal{B}(\mathcal{C}(R'),T,\mathbf{v})}$; /* modified alg. 2 */
12 $\quad$ **repeat**
13 $\quad\quad \mathbf{p}_{old} \leftarrow \mathbf{p}$;
14 $\quad\quad$ **foreach** $s \in S \setminus T$ **do**
15 $\quad\quad\quad \mathbf{a}(s) \leftarrow \arg\min_{a \in A} SV[\mathbf{s}](\mathbf{p}_{old}, s, a)$;
16 $\quad\quad\quad \mathbf{p}(s) \leftarrow \min_{a \in A} SV[\mathbf{s}](\mathbf{p}_{old}, s, a)$;
17 $\quad\quad \mathbf{p} \leftarrow [\![\, \mathbf{p} \,]\!]^{cap}_{R'}$;
18 $\quad\quad$ **foreach** $s \in S \setminus T$ **do**
19 $\quad\quad\quad$ **if** $\mathbf{p}(s) < \mathbf{p}_{old}(s)$ **then**
20 $\quad\quad\quad\quad \Phi(s)(\mathbf{p}(s)) \leftarrow \mathbf{a}(s)$;
21 $\quad$ **until** $\mathbf{p}_{old} = \mathbf{p}$;
22 $\quad \mathit{Unusable} \leftarrow \{r \in R' \mid \mathbf{p}(r) > cap\}$;
23 **until** $\mathit{Unusable} = \emptyset$;
24 **return** $\mathbf{p}, \Phi$

a property of practical importance. For example, we expect that a patrolling unmanned vehicle visits all the checkpoints in a reasonable amount of time. The proposed algorithms are purely qualitative without any consideration for the number of steps. To address this shortcoming, this section proposes two heuristics that can improve ERT: the goal-leaning heuristic and the threshold heuristic. These heuristics modify the proposed algorithms to ensure that the strategies can often hit T sooner than the strategies produced by the unmodified algorithms.

A. Expected reachability time

To formally define ERT, we introduce a new objective: F^i_T (reachability first in i steps) as $\mathsf{F}^i_T = \{{}^d\varrho \in \mathsf{R}^i_T \mid {}^d\varrho \notin \mathsf{R}^j_T \text{ for all } 0 \leq j < i\}$ (the set of all safe loaded runs ${}^d\varrho$ such that the minimum j such that $\varrho_j \in T$ is equal to i). Finally, the expected reachability time for a strategy σ, an initial state $s \in S$, an initial load $d \leq cap$, and a target set $T \subseteq S$ is defined as follows:

$$ERT_{\mathcal{C}}(\sigma, s, d, T) = \sum_{i \in \mathbb{N}} i \cdot {}^d_s\mathbb{P}^{\sigma}_{\mathcal{C}}(\mathsf{F}^i_T). \tag{10}$$

The running example for this section is the CMDP in Fig. 5 with capacity ≥ 3 and the almost-sure satisfaction of the reachability objective for $T = \{w\}$. Consider the following two memoryless strategies that differ only in the action played in t: σ_{a} always plays a in t and σ_{b} always plays b. Loaded with 2 units of resource in t, we have $ERT(\sigma_{\mathsf{a}}, t, 2, \{w\}) = 2$ and $ERT(\sigma_{\mathsf{b}}, t, 2, \{w\}) = 20$. Indeed, σ_{a} surely reaches w in 2 steps while σ_{b} needs 10 trials on average before reaching w via v, each trial needing 2 steps before coming back to t.

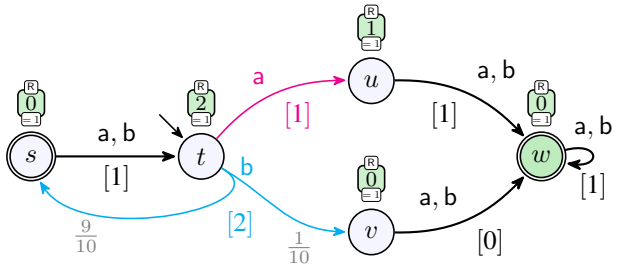


Fig. 5: Example CMDP for the goal-leaning heuristic. The green boxes above states indicate $\mathbf{ml}[\mathsf{R}_{\{w\}}]^{=1}$.

B. Goal-leaning heuristic

Actions to play in certain states with a particular amount of resource are selected on Line 9 of Algorithm 3 (and on Line 15 of Algorithm 6) based on the actions' save values SV. This value in t based on $\mathbf{ml}[\mathsf{R}]^{=1}$ is equal to 2 for both actions a and b. Thus, Algorithm 3 (and also Algorithm 6) returns σ_{a} or σ_{b} randomly based on the resolution of the $\arg\min$ operator on Line 9 (Line 15 in Algorithm 6) for t. The goal-leaning heuristic fixes the resolution of the $\arg\min$ operator to always pick a in this example.

The ordinary $\arg\min$ operator selects randomly an action from the pool of actions with the minimal value $v_{\min}$ of the function SV for t (and the current values of $\mathbf{p}_{old}$). Loosely speaking, the goal-leaning $\arg\min$ operator chooses, instead, the action whose chance to reach the desired successor used to obtain $v_{\min}$ is maximal among actions in this pool.

The value SV is computed using successors' hope values (HV), see Section V. The goal-leaning $\arg\min$ operator records, when computing the HV values, also the transition probabilities of the desired successors. Let s be a state, let a be an action, and let $s' \in Succ(s, a)$ be the successor of a in s that minimizes $HV(\mathbf{p}_{old}, s, a, s')$ and maximizes $\Delta(s, a, s')$ (in this order). We denote by $p_{s,a}$ the value $\Delta(s, a, s')$. The goal-leaning $\arg\min$ operator chooses the action a in s that minimizes $SV(\mathbf{p}_{old}, s, a)$ and maximizes $p_{s,a}$.

In the example from Fig. 5 we have that $p_{t,\mathsf{a}} = 1$ and $p_{t,\mathsf{b}} = \frac{1}{10}$ (as v is the desired successor) in the second iteration of the repeat-loop on Lines 6 to 15. In the last iteration, $p_{t,\mathsf{a}}$ remains 1 and $p_{t,\mathsf{b}}$ changes to $\frac{9}{10}$ as the desired successor changes to s. In both cases, a is chosen by the goal-leaning $\arg\min$ operator as $p_{t,\mathsf{a}} > p_{t,\mathsf{b}}$.

Correctness. We have only changed the behavior of the $\arg\min$ operator when multiple candidates could be used. The correctness of our algorithms does not depend on this choice and thus the proofs apply also to the variant with the goal-leaning operator.

While the goal-leaning heuristic is simple, it has a great effect in practical benchmarks; see Section IX. However, there

are scenarios where it still fails. Consider now the CMDP in Fig. 6 with capacity at least 3. Note that now $\gamma(t, \mathsf{b})$ equals to 1 instead of 2. In this case, even the goal-leaning heuristic prefers b to a in t whenever the current resource level is at least 1 as $SV(\mathbf{p}_{old}, t, \mathsf{b}) = 1 < 2 = SV(\mathbf{p}_{old}, t, \mathsf{a})$ from the second iteration of the repeat-loop onward.

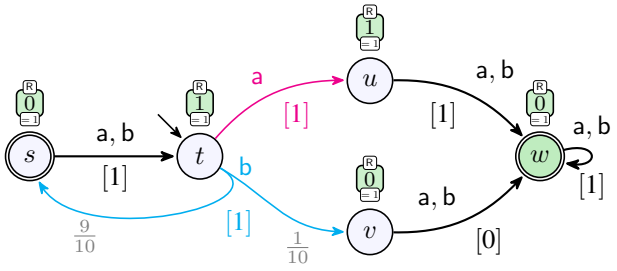


Fig. 6: Example CMDP for the threshold heuristic. In comparison to Fig. 5, the consumption of b is 1 instead of 2.

Note also that the strategy σ_{a} that always plays a in t is not a witness strategy for $\mathbf{ml}[\mathsf{R}]^{=1}$ as σ_{a} needs at least 2 units of resource in t. The desired strategy π should behave in t as follows: play a if the current resource is at least 2 and otherwise play b. We have that $ERT(\pi, t, 2, \{w\}) = 2$ and $ERT(\pi, t, 1, \{w\}) = 3.8$. In the next section, we extend the goal-leaning heuristic to produce π for the running example.

C. Threshold heuristic

The threshold heuristic is parametrized by a probability threshold $0 \leq \theta \leq 1$. Intuitively, when we compute the value of SV for b in t, we ignore the hope values of successors $t' \in Succ(t, \mathsf{b})$ such that $\Delta(t, \mathsf{b}, t') < \theta$. With $\theta = 0.2$, v in our example is no longer considered as a valid outcome for b in t in the second iteration and $SV(\mathbf{p}_{old}, t, \mathsf{b}) = \infty$. Therefore a is picked with $SV(\mathbf{p}_{old}, t, \mathsf{a}) = 2$. It happens only in the fourth iteration that action b is considered from t. In this iteration, $\mathbf{p}_{old}(s)$ is 0 (it is a reload state) and with $\Delta(t, \mathsf{b}, s) = 0.9$, s passes the threshold and we finally have that $SV(\mathbf{p}_{old}, t, \mathsf{b}) = 1$. The resulting finite counter strategy is exactly the desired strategy π from above.

Formally, we parametrize the function SV by θ as follows where we assume min of the empty set is equal to ∞ (changes to definition of SV are highlighted in red):

$$SV_\theta(\mathbf{x}, s, a) = \gamma(s, a) + \min_{\substack{s' \in Succ(s,a) \\ \Delta(s,a,s') \geq \theta}} HV(\mathbf{x}, s, a, s'). \quad (11)$$

The new function SV_θ is a generalization of $SV = SV_0$. To implement this heuristic, we need, in addition to the goal-leaning $\arg\min$ operator, to use SV_θ instead of SV in Algorithms 3 and 6.

There is, however, still one caveat introduced by the threshold. By ignoring some outcomes, the threshold heuristic might compute only over-approximations of $\mathbf{ml}[\mathsf{R}]^{>0}$. As a consequence, the strategy σ computed by the heuristic might be incomplete; it might be undefined for a resource level from which the objective is still satisfiable.

In order to make σ complete and to compute $\mathbf{ml}[\mathsf{R}]^{>0}$ precisely, we continue with the iterations, but now using SV_0 instead of SV_θ. To be more precise, we include Lines 6 to 15 in Algorithm 3 twice (and analogously for Algorithm 6), once with SV_θ and once with SV_0 (in this order).

This extra fixed-point iteration can complete σ and improve $\mathbf{p}$ to match $\mathbf{ml}[\mathsf{R}]^{>0}$ using the rare outcomes ignored by the threshold. As a result, σ behaves according to the threshold heuristic for sufficiently high resource levels and, at the same time, it achieves the objective from every state-level pair where this is possible.

Correctness. The function SV_θ clearly over-approximates SV as we restrict the domain of the min operator only. The invariant of the repeat-loop from the proof of Theorem 6 still holds even when using SV_θ instead of SV (it also obviously holds in the second loop with SV_0). The extra repeat-loop with SV_0 converges to the correct fixed point due to the monotonicity of $\mathbf{p}$ over iterations. Thus, Theorems 5 and 6 hold even when using the threshold heuristics.

D. Limitations

The suggested heuristics naturally do not always produce strategies with the least ERT possible for given CMDP, state, and initial load. Consider the CMDP in Fig. 7 with capacity at least 2. Both heuristics prefer (regardless θ) b in s since $\Delta(s, \mathsf{b}, v) > \Delta(s, \mathsf{a}, v) = \Delta(s, \mathsf{a}, u)$. Such strategy yields ERT from s equal to $2\frac{2}{3}$, while the strategy that plays a in s comes with ERT equal to 2.

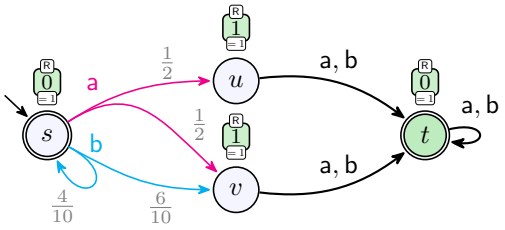


Fig. 7: CMDP illustrating limitations of the goal-leaning heuristics. All actions consume 1 unit of resource.

This non-optimality must be expected as the presented algorithm is purely qualitative. However, there is no known polynomial (with respect to the CMDP representation) algorithm for quantitative analysis of CMDPs that we could use here instead of our approach.

While other, perhaps more involved heuristics might be invented to solve some particular cases, qualitative algorithms which do not track precise values of ERT, naturally cannot guarantee optimality with respect to ERT. The presented heuristics are designed to be simple (both in principle and computation overhead) and to work well on systems with rare undesired events.

The threshold heuristic relies on a well-chosen threshold θ that needs to be provided by the user. Typically, θ should be chosen to be higher than the probability of the most common rare events in the model, to work well. As the presented algorithms rely on the fact that the whole model is known, a suitable threshold might be automatically inferred from the model.

Despite these limitations, we show the utility of the presented heuristics on a case study in the next section.

IX. Implementation and evaluation

We implemented Algorithms 1 to 6, including the proposed heuristics in a tool called FiMDP (Fuel in MDP). The rest of this section provides an overview of all the tools used, an illustrative example, scalability studies, and numerical experiments that demonstrate the utility of CMDP framework.

A. Tools and environments

We utilize FiMDP [25], Storm [7] and FiMDPEnv [26] for implementation and evaluation of our algorithms. FiMDP is an open-source Python library for CMDPs that supports integration with interactive Jupyter notebooks. Storm is an open-source, state-of-the-art probabilistic model checker designed to be efficient in both time and memory. FiMDPEnv is an open-source library for simulating real-world stochastic resource-constrained problems and supports high-level planning tasks in the unmanned underwater vehicle (UUV) and the autonomous electric vehicle (AEV) environments.

The standard UUV environment in FiMDPEnv models the high-level dynamics of UUVs operating in stochastic ocean currents. Each cell in the discretized state space forms one state in the corresponding CMDP, some of which are reload states, and some others form the set of targets T. The set of actions consists of two classes of actions: (i) weak actions that consume less energy but have stochastic outcomes, and (ii) strong actions that have deterministic outcomes with the downside of significantly higher resource consumption. For each class, the environment offers up to 8 directions (east, north-east, north, north-west, west, south-west, south, and south-east). The stochastic dynamics are generated using the ocean current information as described in [27].

The AEV environment in FiMDPEnv models the routing problem of an autonomous electric vehicle operating in the streets of Manhattan, New York using real-world energy consumption data. Intersections in the street network and directions of feasible movement form the state and action spaces of the MDP. We use intersections in the proximity of real-world fast-charging stations as the set of reload states.

B. Strategy synthesis for CMDPs in FiMDP and Storm

We demonstrate the efficiency of the CMDP formulation using 15 strategy synthesis tasks with a Büchi objective generated on the UUV environment. The complexity of a task in this environment is determined by the grid size and the capacity. We use grid sizes 10, 20, and 50. For each grid size n, we create five tasks with capacities equal to 1, 2, 3, 5, and 10 times n. We solve each task modeled as a CMDP using FiMDP and modeled as a regular MDP with resource constraints encoded in states and actions using Storm. We express the qualitative Büchi property in PCTL [28] for Storm. Figure 8 presents the running times (averaged over 10 independent runs) needed for each task by FiMDP (•) and by Storm (×).

We can observe that FiMDP outperforms Storm in terms of computation time in all test cases with the exception of small problems. For the small tasks, Storm benefits from its efficient implementation in C++. The advantage of FiMDP lies in the fact that the state space of CMDPs (and also the time needed for their analysis) does not grow with rising capacity.

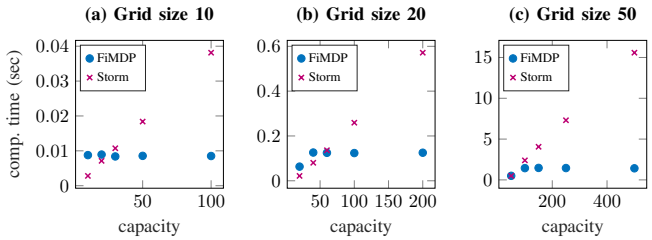


Fig. 8: Mean computation times for solving the CMDP model of the UUV environment with capacities proportional to the grid size in each task. Each subplot in the figure corresponds to a different size of the grid-world.

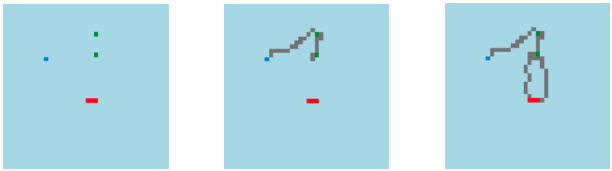
C. Patrolling problem in the UUV environment

While the standard UUV environment presented in Section IX-A includes deterministic strong actions, we note that this is not essential and one can model real-world problems with only stochastic weak actions. This is possible by (i) restricting the stochastic outcome of the weak action to the intended outcome and the neighboring cells, and (ii) defining reload states spanning three primitive cells in the grid-world.

We provide an illustrative example by considering the scenario of a UUV patrolling two targets in an area with one reload base. We model the problem using the UUV environment without strong actions and synthesize our strategy using almost-sure Büchi objective with the threshold heuristic described in Section VIII. Fig. 9 shows a trajectory obtained by following the synthesized strategy where the agent patrols both the points of interest constantly while also reloading at the base to prevent running out of energy. Table I summarizes the expected reachability time (ERT) for different strategies and grid sizes. The ERT values are calculated by running 10000 simulations with each strategy for 500 steps and taking the average over the reachability time from the simulations.

TABLE I: ERT for strategies on the patrolling problem

operator	θ	ERT - 40 grid size	ERT - 80 grid size
default solver	—	500+	500+
goal-leaning heuristic	—	46.44	83.38
threshold heuristic	0.3	22.27	43.54
threshold heuristic	0.5	35.52	51.86



(a) $t, e = 0, 50$ (b) $t, e = 24, 26$ (c) $t, e = 50, 39$

Fig. 9: Illustration showing the trajectory of a UUV patrolling the points of interest at different time instances (t) and energy levels (e). The grid-world shown is of size 40 with an initial state, a reload state, and two target states.

We now proceed to detailed analysis of different heuristic algorithms using both UUV and AEV environments.

D. Heuristics for improving ERT

This section investigates the proposed heuristics from a practical, optimal decision-making perspective. We consider two testing scenarios with UUV and AEV environments.

1) UUV Environment: We consider the standard UUV environment with a grid size of 20, a single reload state and one target state. The objective of the agent is to reach the target almost-surely. We consider four strategies generated for almost-sure reachability by the following algorithms: the standard strategy (using default randomized $\arg\min$ operator), goal-learning strategies (goal-leaning $\arg\min$ operator), and threshold strategies with thresholds θ equal to 0, 0.3 and 0.5. Figure 10 illustrates the UUV behavior while following different strategies starting with an energy level of 30.

We calculate the approximate ERT for each of the four strategies by averaging over 10000 independent runs. The strategy built using the standard $\arg\min$ operator, as apparent from 10, does not reach the target within the first 200 steps in any of the 10000 trials. The goal-leaning $\arg\min$ operator itself helps a lot to navigate the agent towards the goal and only takes about 26.45 steps to reach the target. However, it still relies on rare events in some places. Setting $\theta = 0.3$ helps to avoid these situations as the unlikely outcomes are not considered anymore thereby reducing the ERT to about 18.34 steps, and finally, $\theta = 0.5$ forces the agent to use strong actions almost exclusively leading to an ERT of about 15 steps – the same as the reachability time of the shortest path assuming deterministic dynamics (see 10). While using thresholds led to a better ERT in this particular environment, the result might not hold in general.

2) AEV Environment: This example considers an autonomous electric vehicle (AEV) tasked with reaching a destination in the shortest amount of time without resource exhaustion. Assigning the destination as the target state and the recharge stations as the reload states, we synthesize strategies using the standard solver and the threshold heuristic and terminate our simulations once we reach the target state.

Figure 11 illustrates the area considered in the AEV environment and shows the trajectories of the AEV from simulation instances while following the synthesized strategies. The agent following the standard solver strategy, with an ERT of 74, relies on rare events at some states and has to visit a reload state before reaching the target. On the other hand, the threshold heuristic strategy with an ERT of 60 results in the agent directly heading to the target state.

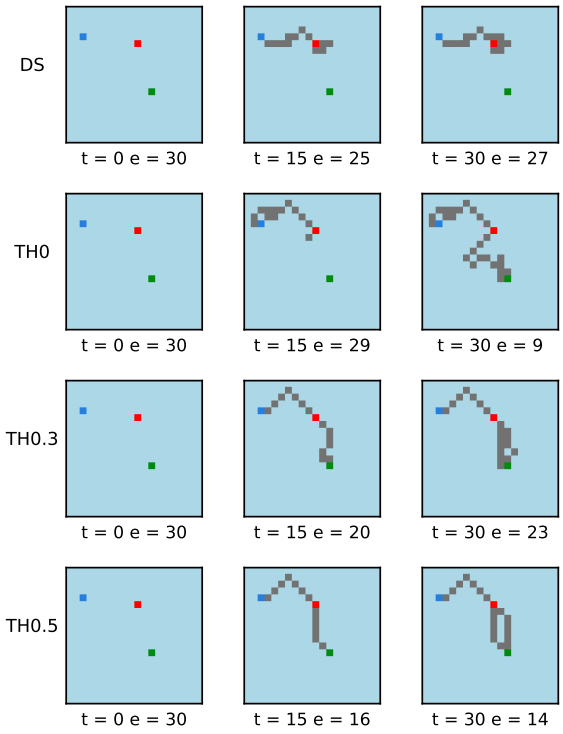
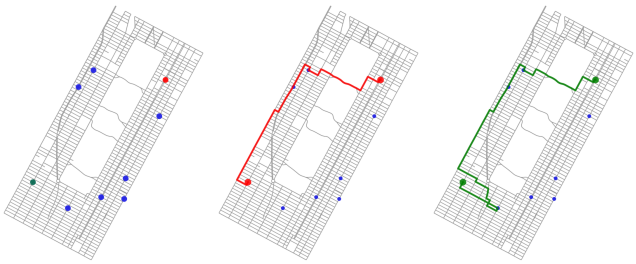


Fig. 10: Comparison of different strategy synthesis algorithms – DS denotes the standard default solver, TH0 denotes the goal-leaning heuristic, TH0.3 denotes threshold heuristic with a threshold of 0.3, and TH0.5 denotes threshold heuristic with a threshold of 0.5.



(a) Map (b) ERT $= 60$ (c) ERT$= 74$

Fig. 11: Illustration showing the AEV environment and the obtained trajectories. (a) street network in the AEV environment with an initial state, multiple reload states, and a target state; (b) trajectory and ERT for the standard solver; (c) trajectory and ERT for the threshold heuristic with $\theta = 0.2$.

X. Conclusion & Future Work

We presented consumption Markov decision processes — models for stochastic environments with resource constraints — and we showed that strategy synthesis for qualitative objectives is efficient. In particular, our algorithms that solve synthesis for almost-sure reachability and almost-sure Büchi objective in CMDPs, work in time polynomial with respect to the representation of the input CMDP. In addition, we presented two heuristics that can significantly improve the expected time needed to reach a target in realistic examples. The experimental evaluation of the suggested methods confirmed that direct analysis of CMDPs in our tool is faster than analysis of an equivalent MDP even when performed by the state-of-the-art tool Storm(with the exception of very small models).

Possible directions for future work include extensions to quantitative analysis (e.g. minimizing the expected resource consumption or reachability time), stochastic games, or partially observable setting.

Acknowledgements: We acknowledge the kind help of Tomáš Brázdil, Vojtěch Forejt, David Klaška, and Martin Kučera in the discussions leading to this paper.

REFERENCES

[1] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1259–1274, 2011.

[2] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2386–2392, 2021.

[3] V. D. Blondel and J. N. Tsitsiklis, "Complexity of stability and controllability of elementary hybrid systems," *Automatica*, vol. 35, no. 3, pp. 479–489, 1999.

[4] P. C. Bell, S. Chen, and L. Jackson, "On the decidability and complexity of problems for restricted hierarchical hybrid systems," *Theoretical Computer Science*, vol. 652, pp. 47–63, 2016.

[5] H. A. Blom, G. Bakker, and J. Krystul, "Probabilistic reachability analysis for large scale stochastic hybrid systems," in *46th Conference on Decision and Control*, 2007, pp. 3182–3189.

[6] M. Prandini and J. Hu, "A stochastic approximation method for reachability computations," in *Stochastic Hybrid Systems*. Springer, 2006, pp. 107–139.

[7] C. Hensel, S. Junges, J.-P. Katoen, T. Quatmann, and M. Volk, "The probabilistic model checker storm," 2020.

[8] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and M. Stoelinga, "Resource interfaces," in *3rd International Workshop on Embedded Software*, 2003, pp. 117–133.

[9] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and J. Srba, "Infinite runs in weighted timed automata with energy constraints," in *6th International Conference on Formal Modelling and Analysis of Timed Systems*, 2008, pp. 33–47.

[10] U. Boker, T. A. Henzinger, and A. Radhakrishna, "Battery transition systems," in *41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2014, pp. 595–606.

[11] G. Bacci, P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and P.-A. Reynier, "Optimal and robust controller synthesis," in *22nd International Symposium on Formal Methods*, 2018, pp. 203–221.

[12] E. R. Wognsen, R. R. Hansen, K. G. Larsen, and P. Koch, "Energy-aware scheduling of FIR filter structures using a timed automata model," in *19th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2016.

[13] G. Sugumar, R. Selvamuthukumaran, T. Dragicevic, U. Nyman, K. G. Larsen, and F. Blaabjerg, "Formal validation of supervisory energy management systems for microgrids," in *43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 1154–1159.

[14] N. Fijalkow and M. Zimmermann, "Cost-parity and cost-Streett games," in *32nd Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2012, pp. 124–135.

[15] C. Baier, C. Dubslaff, S. Klüppelholz, and L. Leuschner, "Energy-utility analysis for resilient systems using probabilistic model checking," in *35th International Conference on Application and Theory of Petri Nets and Concurrency*, 2014, pp. 20–39.

[16] C. Baier, M. Daum, C. Dubslaff, J. Klein, and S. Klüppelholz, "Energy-utility quantiles," in *6th International Symposium on NASA Formal Methods*, 2014, pp. 285–299.

[17] K. Chatterjee and L. Doyen, "Energy and mean-payoff parity Markov decision processes," in *Proceedings of MFCS 2011*, vol. 6907, 2011, pp. 206–218.

[18] M. Jurdziński, "Deciding the winner in parity games is in UP ∩ co-UP," *Information Processing Letters*, vol. 68, no. 3, pp. 119–124, 1998.

[19] T. Brázdil, K. Chatterjee, A. Kučera, and P. Novotný, "Efficient controller synthesis for consumption games with multiple resource types," in *Proceedings of CAV 2012*, vol. 7358, 2012, pp. 23–38.

[20] F. Blahoudek, T. Brázdil, P. Novotný, M. Ornik, P. Thangeda, and U. Topcu, "Qualitative controller synthesis for consumption Markov decision processes," in *32nd International Conference on Computer-Aided Verification*, vol. II, 2020, pp. 421–447.

[21] R. Ash and C. Doléans-Dade, *Probability and Measure Theory*. Academic Press, 2000.

[22] C. Courcoubetis and M. Yannakakis, "The Complexity of Probabilistic Verification," *J. ACM*, vol. 42, no. 4, pp. 857–907, 1995.

[23] K. R. Apt and E. Grädel, *Lectures in Game Theory for Computer Scientists*, 1st ed. USA: Cambridge University Press, 2011.

[24] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao, "On short paths interdiction problems: total and node-wise limited interdiction," *Theory of Computing Systems*, vol. 43, no. 2, pp. 204–233, 2008.

[25] "FiMDP - Fuel in Markov Decision Processes," https://github.com/FiMDP/FiMDP, 2021.

[26] "FiMDPEnv - Environments for FiMDP," https://github.com/FiMDP/FiMDPEnv, 2021.

[27] W. H. Al-Sabban, L. F. Gonzalez, and R. N. Smith, "Extending persistent monitoring by combining ocean models and Markov decision processes," in *2012 Oceans*. IEEE, 2012, pp. 1–10.

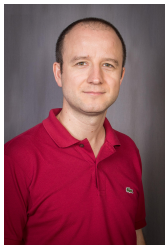
[28] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

František Blahoudek is with Pure Storage, Prague, Czech Republic. He was a postdoctoral researcher at the Faculty of Information Technology, Brno University of Technology, Czech Republic and a postdoctoral researcher in the group of Ufuk Topcu at the University of Texas at Austin. He received his Ph.D. degree from the Masaryk University, Brno in 2018. His research focuses on automata in formal methods and on planning under resource constraints.

Petr Novotný is an assistant professor at the Faculty of Informatics, Masaryk University, Czech Republic. He received his Ph.D. degree from Masaryk University in 2015. His research focuses on automated analysis of probabilistic program, application of formal methods in the domains of planning and reinforcement learning, and on the theoretical foundations of probabilistic verification.

Melkior Ornik is an assistant professor in the Department of Aerospace Engineering and the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign. He received his Ph.D. degree from the University of Toronto in 2017. His research focuses on developing theory and algorithms for learning and planning of autonomous systems operating in uncertain., complex and changing environments, as well as in scenarios where only limited knowledge of the system is available.

Pranay Thangeda is a graduate student in the Department of Aerospace Engineering and the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign. He received his M.S. degree in Aerospace Engineering from the University of Illinois at Urbana-Champaign in 2020. His research focuses on developing algorithms that exploit side information for efficient planning and learning in unknown environments.



Ufuk Topcu is an associate professor in the Department of Aerospace Engineering and Engineering Mechanics and the Oden Institute at The University of Texas at Austin. He received his Ph.D. degree from the University of California at Berkeley in 2008. His research focuses on the theoretical, algorithmic, and computational aspects of design and verification of autonomous systems through novel connections between formal methods, learning theory and controls.

APPENDIX

A. Proof of Lemma 3

Proof of Lemma 3. We proceed by induction on i. The base case is clear. Now assume that the statement holds for some $i \geq 0$. Fix any s. Denote by $b = \mathcal{B}^{i+1}(\mathbf{y}_T)(s)$ and $d = \mathbf{ml}[\mathsf{R}^{i+1}]^{>0}(s)$. We show that $b = d$. The equality trivially holds whenever $s \in T$, so in the remainder of the proof we assume that $s \notin T$.

We first prove that $b \geq d$. If $b = \infty$, this is clearly true. Otherwise, let $a_{\min}$ be the action minimizing $SV(\mathcal{B}^i(\mathbf{y}_T), s, a_{\min})$ (which equals b if $s \notin R$) and let $t_{\min} \in Succ(s, a_{\min})$ be the successor with the lowest hope value. We denote by p the value $\mathbf{ml}[\mathsf{R}^i]^{>0}(t_{\min}) \geq \mathbf{ml}[\mathsf{S}](t_{\min})$ in the following two paragraphs. By induction hypothesis, there exists a strategy σ_1 such that $\sigma_1 \, {}^{p}_{t_{\min}}\!\models^{>0} \mathsf{R}^i$, and there also exists a strategy σ_2 such that $\sigma_2 \, {}^{l}_{t}\!\models \mathsf{S}$ for all other successors $t \in Succ(s, a_{min}), t \neq t_{\min}$ with $l = \mathbf{ml}[\mathsf{S}](t)$. We now fix a run ${}^{p}\rho$ as a run from $\mathsf{Comp}(\sigma_1, t_{\min}, p)$ that reaches T in at most i steps (which must exists).

Consider now a strategy π which, starting in s, plays $a_{\min}$. If the outcome of $a_{\min}$ is $t_{\min}$, the strategy π starts to mimic σ_1, otherwise it starts to mimic σ_2. By definition of SV we have that $b \geq \gamma(s, a_{\min}) + l$ for $l = \mathbf{ml}[\mathsf{S}](t)$ for all $t \in Succ(s, a_{\min})$ (including $t_{\min}$) and thus $\pi \, {}^{b}_{s}\!\models \mathsf{S}$. The loaded run ${}^{b}s a_{\min} t_{\min} \odot \rho \in \mathsf{Comp}(\pi, s, b)$ is safe (as $b \geq \gamma(s, a_{\min}) + p$) and thus it is the witness that $\pi \, {}^{b}_{s}\!\models^{>0} \mathsf{R}^{i+1}$.

Now we prove that $b \leq d$. This clearly holds if $d = \infty$, so in the remainder of the proof we assume $d \leq cap$. By the definition of d there exists a strategy σ such that $\sigma \, {}^{d}_{s}\!\models^{>0} \mathsf{R}^{i+1}$ Let $a = \sigma({}^{d}s)$ be the action selected by σ in the first step when starting in s loaded by d. For each $t \in Succ(s, a)$ we assign a number d_t defined as $d_t = 0$ if $t \in R$ and $d_t = lastRL({}^{d}sat) = d - \gamma(s, a)$ otherwise.

We finish the proof by proving these two claims:
(1) It holds $SV(\mathcal{B}^i(\mathbf{y}_T), s, a) \leq \gamma(s, a) + \max_{t \in Succ(s,a)} d_t$.
(2) If $s \notin R$, then $\gamma(s, a) + \max_{t \in Succ(s,a)} d_t \leq d$.
Let us first see why these claims are indeed sufficient. From (1) we get $\mathcal{A}(\mathcal{B}^i(\mathbf{y}_T))(s) \leq \gamma(s, a) + \max_{t \in Succ(s,a)} d_t \leq cap$ (from the definition of $RL({}^{d}sat)$). If $s \in R$, then it follows that $b = [\![\, \mathcal{A}(\mathcal{B}^i(\mathbf{y}_T)) \,]\!]^{cap}_R(s) = 0 \leq d$. If $s \notin R$, then $b = [\![\, \mathcal{A}(\mathcal{B}^i(\mathbf{y}_T)) \,]\!]^{cap}_R(s) = \mathcal{A}(\mathcal{B}^i(\mathbf{y}_T))(s) \leq \gamma(s, a) + \max_{t \in Succ(s,a)} d_t \leq d$, the first inequality shown above and the second coming from (2).

Let us first prove (1.). We denote by τ the strategy such that for all histories α we have $\tau(\alpha) = \sigma(sa\alpha)$. For each $t \in Succ(s, a)$, we have $\tau \, {}^{d_t}_{t}\!\models \mathsf{S}$. Moreover, there exists $q \in Succ(s, a)$ such that $\tau \, {}^{d_q}_{q}\!\models^{>0} \mathsf{R}^i$ (since $s \notin T$); hence, by induction hypothesis it holds $\mathcal{B}^i(\mathbf{y}_T)(q) \leq d_q$. From this and from the definition of SV we get

$$\begin{aligned} SV(\mathcal{B}^i(\mathbf{y}_T), s, a) &\leq \gamma(s, a) + HV(\mathcal{B}^i(\mathbf{y}_T), s, a, q) \\ &\leq \gamma(s, a) + \max_{\substack{t \in Succ(s,a) \\ t \neq q}} \{\mathcal{B}^i(\mathbf{y}_T)(q), \mathbf{ml}[\mathsf{S}](s)\} \\ &\leq \gamma(s, a) + \max_{t \in Succ(s,a)} d_t. \end{aligned}$$

To finish, (2) follows immediately from the definition of d_t and the fact that $lastRL({}^{d}sat)$ is always bounded from above by $d - \gamma(s, a)$ for $s \notin R$. □

B. Proof of Lemma 4

Proof of Lemma 4. By Lemma 3, it suffices to show that $\mathbf{ml}[\mathsf{R}]^{>0} = \mathbf{ml}[\mathsf{R}^K]^{>0}$. To show this, fix any state s such that $\mathbf{ml}[\mathsf{R}]^{>0}(s) < \infty$. For the sake of succinctness, we denote $\mathbf{ml}[\mathsf{R}]^{>0}(s)$ by d. To each strategy π such that $\pi \, {}^{d}_{s}\!\models^{>0} \mathsf{R}$ we assign a reachability indexthat is the infimum of all i such that $\pi \, {}^{d}_{s}\!\models^{>0} \mathsf{R}^i$. Let σ be a strategy such that $\sigma \, {}^{d}_{s}\!\models^{>0} \mathsf{R}$ with the minimal reachability index k. We show that the $k \leq K$.

We proceed by a suitable "strategy surgery" on σ. Let α be a history produced by σ from s of length k whose last state belongs to T. Assume, for the sake of contradiction, that $k > K$. This can only be if at least one of the following conditions hold:
(a) Some reload state is visited twice on α, i.e. there are $0 \leq j < l \leq k + 1$ such that $\alpha_l = \alpha_j \in R$, or
(b) some state is visited twice with no intermediate visits to a reload state; i.e., there are $0 \leq j < l \leq k + 1$ such that $\alpha_j = \alpha_l$ and $\alpha_h \notin R$ for all $j < h < l$.
Indeed, if none of the conditions hold, then the reload states partition α into at most $|R| + 1$ segments, each segment containing non-reload states without repetition. This would imply $k = len(\alpha) \leq K$.

In both cases (a) and (b) we can arrive at a contradiction using essentially the same argument. Let us illustrate the details on case (a): Consider a strategy π such that for every history of the form $\alpha_{..j} \odot \beta$ for a suitable β we have $\pi(\alpha_{..j} \odot \beta) = \sigma(\alpha_{..l} \odot \beta)$; on all other histories, π mimics σ. Clearly $\pi \, {}^{d}_{s}\!\models \mathsf{S}$: the behavior changed only for histories with $\alpha_{..j}$ as a prefix and for each suitable β we have $lastRL({}^{d}\alpha_{..j} \odot \beta) = lastRL({}^{d}\alpha_{..l} \odot \beta)$ due to the fact that $\alpha_j = \alpha_l$ is a reload state. Moreover, we have that ${}^{d}\alpha_{..j} \odot \alpha_{l..k}$ created by π reaches T in $k' = k - (l - j) < k$ steps which is the reachability index of π. We reached a contradiction with the choice of σ.

For case (b), the only difference is that now the resource level after $\alpha_{..j} \odot \beta$ can be higher then the one of $\alpha_{..l} \odot \beta$ due to the removal of the intermediate non-reloading cycle. Since we need to show that the energy level never drops below 0, the same argument works. □

C. Computation Complexity of Algorithms

We present the explicit complexities of the six algorithms (1, 2, 3, 4, 6, 5) in terms of the parameters describing $\mathcal{C}$. Let δ denote the branching factor defined as the maximal number of successors of some state under some action, i.e., $\delta = \max_{(s,a) \in S \times A} |Succ(s, a)|$. Also recall that n denotes the length of the longest simple path in $\mathcal{C}$. Then, the complexities of our algorithms are as follows: Algorithm 1 – $\mathcal{O}(|S||A|\delta n)$, Algorithm 2 – $\mathcal{O}(|R||S||A|\delta n)$, Algorithm 3 – $\mathcal{O}(|R||S||A|\delta(n + \delta|S|)$, Algorithm 4 – $\mathcal{O}(|R|^2|S||A|\delta(n + \delta|S|))$, Algorithm 5 – $\mathcal{O}(|S||A|\delta n)$, and Algorithm 6 – $\mathcal{O}(|R|^2|S||A|\delta(n + \delta|S|))$.