Shielding in Resource-Constrained Goal POMDPs

Michal Ajdarów, Šimon Brlej, Petr Novotný

Faculty of Informatics, Masaryk University
{xajdarow,xbrlej,petr.novotny}@fi.muni.cz

Abstract

We consider partially observable Markov decision processes (POMDPs) modeling an agent that needs a supply of a certain resource (e.g., electricity stored in batteries) to operate correctly. The resource is consumed by the agent's actions and can be replenished only in certain states. The agent aims to minimize the expected cost of reaching some goal while preventing resource exhaustion, a problem we call *resource-constrained goal optimization* (RSGO). We take a two-step approach to the RSGO problem. First, using formal methods techniques, we design an algorithm computing a *shield* for a given scenario: a procedure that observes the agent and prevents it from using actions that might eventually lead to resource exhaustion. Second, we augment the POMCP heuristic search algorithm for POMDP planning with our shields to obtain an algorithm solving the RSGO problem. We implement our algorithm and present experiments showing its applicability to benchmarks from the literature.

Introduction

Partially observable Markov decision processes (POMDPs) are the standard model for decision making under uncertainty. While POMDPs are computationally demanding to solve, advances in heuristic search (Silver and Veness 2010) and reinforcement learning (Bhattacharya et al. 2020) allowed for tackling large POMDP models. Recently, increasing attention is being paid to *safety aspects* of autonomous decision making as opposed to pure optimization of the expected rewards or costs (García and Fernández 2015). Indeed, heuristic and learning techniques can be susceptible to leading the decision-making agent into dangerous situations, and additional care must be taken to *formally guarantee* the absence of such a risky behavior.

Shielding. A promising approach to obtaining such safety guarantees is offered by the concept of *permissive controller synthesis* (Dräger et al. 2015; Junges et al. 2016), which was later distilled into the concept of *shielding* (Alshiekh et al. 2018). Intuitively, a shield σ is a function which inputs the agent's current information X (in a POMDP, this would be the whole history of the agent's actions and observations) and outputs a list of *allowed* actions which the agent can use in the current situation without risking violation of its safety specification in the future. The shielding process then forces the agent to only consider, in a situation described by available information X, the allowed actions in the list $\sigma(X)$. Shields are typically computed via *formal methods* approaches, and hence they can guarantee that the shielded algorithm satisfies the desired safety specification.

Consumption Models and RSGO Problem. In this paper, we focus on safe decision making in agents that require some uninterrupted supply of a resource (such as electricity) to operate correctly. Such agents can be encountered, e.g., in robotics (Notomista, Ruf, and Egerstedt 2018), and we model them via the *consumption* paradigm (Brázdil et al. 2012), where the available amount of the resource is represented by an integer from the set $\{0, 1, \ldots, ca\}$, the number ca denoting the agent's *battery capacity*. Each action of the agent consumes some amount of the resource (i.e., decreases the resource level), and the resource can be replenished to the full capacity only in special *reload states* (e.g., charging stations). The safety specification is that the agent must never run out of the resource. A crucial property of consumption models (such as *consumption MDPs* (Blahoudek et al. 2020)) is that the resource levels are *not* encoded in the model's states (since this would blow up the state space by a factor exponential in the bit-size of ca). Instead, the amount of resource consumed is specified for each state-action pair, and the agent then tracks its resource level itself. To account for the fact that, apart from preventing resource exhaustion, the agent aims to do something useful, we study the *resource-safe goal optimization (RSGO)* problem in the consumption setting: the agent aims to reach a given set of *goal states* with probability 1 (i.e., almost-surely) at the minimal expected cost, while preventing resource exhaustion.

Limitations of Previous Work. The previous approaches to consumption models for resource-constrained decision making under uncertainty suffer from two key limitations: First, they consider only perfectly observable setting, i.e., consumption MDPs. Second, they only consider computing policies satisfying *qualitative criteria:* avoiding resource exhaustion and almost-surely reaching a goal state; optimization of *quantitative criteria,* such as the expected cost of achieving a goal, was not considered.

Our Contribution. In this paper, we overcome both aforementioned limitations: we present a method for solving a combination of qualitative and quantitative criteria in *partially observable* consumption MDPs (CoPOMDPs). Our contribution has two essential parts: First, we show how to design an algorithm computing shields in consumption POMDPs prohibiting exactly those behaviors that lead to resource exhaustion or that violate the possibility of eventually reaching a goal state. Hence, our shields handle the qualitative aspect of the RSGO problem. Second, to handle also the quantitative aspect, we augment the well-known POMCP heuristic planning algorithm (Silver and Veness 2010) with our shields, thus obtaining an algorithm for the finite-horizon approximation of the RSGO problem. We implement our new algorithm and demonstrate its applicability to benchmarks derived from the literature.

Outline of Techniques. The previous work (Blahoudek et al. 2020) presented an algorithm that for *perfectly observable* consumption MDPs (CoMDPs) computes a policy ensuring almost-sure goal reachability while preventing resource exhaustion. The algorithm runs in time polynomial in the size of the CoMDP, so in particular in time polynomial in the bit-size of ca. We reduce the computation of shields for CoPOMDPs to the CoMDP problem of (Blahoudek et al. 2020). The reduction is non-trivial: in particular, we show that the standard technique of constructing a *belief support MDP* (Chatterjee et al. 2016; Junges, Jansen, and Seshia 2021; Baier, Bertrand, and Größer 2008) and then applying the algorithm for perfectly observable MDPs is not directly usable in the consumption setting.

Related Work. Shielding in MDPs and POMDPs was studied w.r.t. state safety specification (avoiding critical states) (Alshiekh et al. 2018), ensuring almost-sure reachability of a goal state (Junges, Jansen, and Seshia 2021), or guaranteeing that the payoff is almost-surely above some threshold (Chatterjee et al. 2017). The related notion of permissive controller synthesis has been studied in more quantitative settings of probabilistic reachability and expected cost (Dräger et al. 2015; Junges et al. 2016). To our best knowledge, our paper is the first to consider shielding for resource-constrained agents modeled via POMDPs. While shielding for CoPOMDPs could be in principle reduced to state safety shielding by encoding resource level into states, this would blow-up the state space by the factor equal to the battery capacity. As demonstrated already for CoMDPs in (Blahoudek et al. 2020), the "resource levels in states" approach is highly inefficient when compared to methods tailored to consumption models.

Resource-constrained planning is particularly relevant in the application domain of autonomous (land-based/aerial/underwater) vehicles, e.g., (Notomista, Ruf, and Egerstedt 2018; Mitchell et al. 2015; Eaton et al. 2018). In formal methods and verification, resource-constrained agents are typically modeled in the consumption framework (used in this paper) or *energy* framework. The former has been considered in both non-probabilistic (Brázdil et al. 2012, 2014) and probabilistic (Blahoudek et al. 2020, 2021) settings, but not in partially observable ones. The energy framework differs from the consumption one by handling reloads: instead of atomic reloads in reload states, a "negative consumption" is enabled for some state-action pairs, allowing for incremental reloading. Various classes of energy models have been considered, e.g., (Chakrabarti et al. 2003; Bouyer et al. 2008; Brázdil, Kučera, and Novotný 2016; Mayr et al. 2017; Degorre et al. 2010), and while they can be seen as more general than consumption models, they are not known to admit algorithms running in time polynomial in the bit-size of the capacity ca.

The constrained optimization aspect present in CoPOMDPs is similar in spirit to *constrained (PO)MDPs* (C(PO)MDPs) (Altman 1999; Undurti and How 2010; Poupart et al. 2015). While both approaches fit the same framework of constrained optimization, the types of constraints are actually quite different. In particular, Co(PO)MDPs *cannot* be viewed as a special case of C(PO)MDPs, and vice versa. The key difference is that in C(PO)MDPs, there are penalty functions determining one-step penalties, and the constraint is that the *expected aggregated* penalty is below a given threshold. (Aggregation functions such as discounted, total, or mean payoff are typically considered.) On the other hand, the constraint in Co(PO)MDPs is on the *intermediate* values of the consumption, not on the expectation of its aggregate values. C(PO)MDPs can impose constraints such as "the expected total amount of the resource consumed by the agent is $\leq B$," which does not guarantee that the agent does not run out of the resource between two reloads. Hence, the two models are incomparable.

Preliminaries

We denote by $\mathcal{D}(X)$ the set of all probability distributions over an at most countable set X and by $supp(d)$ the support of a distribution d. For $n \in \mathbb{N}$ we denote by $[n]$ the integer interval $\{0, 1, \ldots, n\} \cup \{\bot\}$, where $\bot$ is a special element deemed smaller than all integers.

Definition 1 (CoPOMDPs.)**.** *A* consumption partially observable Markov decision process (CoPOMDP) *is a tuple* $\mathcal{C} = (S, A, \delta, \mathcal{Z}, \mathcal{O}, C, R, ca)$*, where* S *is a finite set of* states*,* A *is a finite set of* actions*,* $\delta : S \times A \to \mathcal{D}(S)$ *is a* probabilistic transition function *that given a state* s *and an action* $a \in A$ *gives the probability distribution over the successor states,* $\mathcal{Z}$ *is a finite set of* observations*,* $\mathcal{O} : S \to \mathcal{D}(\mathcal{Z})$ *is a probabilistic* observation function *that maps every state to a distribution over observations,* $C \colon S \times A \to \mathbb{N}$ *is a* resource consumption function*,* $R \subseteq S$ *is the set of* reload states*, and* $ca \in \mathbb{N}^+$ *is a* resource capacity.

We often abbreviate $\delta(s, a)(s')$ *and* $\mathcal{O}(s)(o)$ *by* $\delta(s'|s, a)$ *and* $\mathcal{O}(o|s)$*, respectively.*

For $(s, a) \in S \times A$ we denote by $Succ(s, a)$ the set of successor states of s under a, i.e., the support of $\delta(s, a)$. We say that two states $s, t \in S$ are *lookalikes* if they can produce the same observation, i.e., if $supp(\mathcal{O}(s)) \cap supp(\mathcal{O}(t)) \neq \emptyset$.

Dynamics of CoPOMDPS. A CoPOMDP $\mathcal{C}$ evolves in discrete time steps. The situation at time $t \in \mathbb{N}$ is described by a pair of random variables, S_t and L_t, denoting the current state and the current resource level at time t, respectively. The agent cannot observe the state directly, instead receiving an *observation* $O_t \in \mathcal{Z}$ sampled according to the current state and $\mathcal{O}$: $O_t \sim \mathcal{O}(S_t)$. The initial state is given by the *initial distribution* $\lambda_0 \in \mathcal{D}(S)$, i.e., $S_0 \sim \lambda_0$, while L_0 will be typically fixed to a concrete *initial resource level* $\ell_0 \in [ca]$. Then, in every step t, the agent selects an action A_t from A according to a *policy* π. The policy makes a (possibly randomized) decision based on the current *history* $H_t = O_0 L_0 A_0 O_1 L_1 A_1 \dots O_{t-1} L_{t-1} A_{t-1} O_t L_t$, a finite alternating sequence of hitherto witnessed observations, resource levels, and actions; i.e., $A_t \sim \pi(H_t)$. The new state S_{t+1} is then sampled according to the transition function δ: $S_{t+1} \sim \delta(S_t, A_t)$. To describe the resource dynamics, we define a *resource update function* $\Delta\colon [ca] \times S \times A \to [ca]$ s.t. $\Delta(\ell, s, a)$ denotes the resource level after making a step from s using action a, provided that the previous resource level was ℓ: if $\ell = \bot$, then $\Delta(\ell, s, a) = \bot$ and otherwise

$$\Delta(\ell, s, a) = \begin{cases} \ell - C(s,a) & \text{if } 0 \leq \ell - C(s,a) \wedge s \notin R, \\ ca - C(s,a) & \text{if } 0 \leq ca - C(s,a) \wedge s \in R, \\ \bot & \text{otherwise.} \end{cases}$$

We then put $L_{t+1} = \Delta(L_t, S_t, A_t)$.

When referring to CoPOMDP dynamics, we use uppercase to denote random variables (S_t, O_t, A_t, etc.) and lowercase for their concrete values, e.g., h_t for a concrete history of length t. We denote by *Hist* the set of all possible histories in a given CoPOMDP, and by $len(h)$ a *length* of a history h, i.e., the number of action appearances in h. We denote by ℓ_h the last resource level of history h.

Assumptions: Observable Resource and Zero Cycles. Including resource levels in histories amounts to making the levels perfectly observable. This is a reasonable assumption, as we expect a resource-constrained agent to be equipped with a resource load sensor. As long as this sensor has at least some guaranteed accuracy ε, for an observed resource level L we can report $\lfloor L - \varepsilon \rfloor$ to the agent as a conservative estimate of the resource amount. We also assume that the agent can recognize whether it is in a reload state or not, i.e., a reload state is never a lookalike of a non-reload state. Finally, we assume that the CoPOMDP does not allow the agent to indefinitely postpone consuming a positive amount of a resource unless a goal state (see below) has already been reached. This is a technical assumption which does not impede applicability, since we typically model autonomous agents whose *every* action consumes some resource.

Optimization in Goal CoPOMDPs. In what follows, we denote by $\mathbb{P}^\pi$ the probability measure induced over the trajectories of a CoPOMDP by a policy π, and by $\mathbb{E}^\pi$ the corresponding expectation operator.

In a *goal CoPOMDP,* we are given a set of *goal states* $G \subseteq S$. A policy π is a *positive-goal* policy if there exists $\epsilon > 0$ such that it reaches a goal state with probability at least ϵ:

$$\mathbb{P}^\pi(S_t \in G \text{ for some } t \in \mathbb{N}) \geq \epsilon,$$

and a *goal* policy if it is a positive-goal policy with $\epsilon = 1$.

We assume that all goal states $g \in G$ are absorbing, i.e., $\delta(g \mid g, a) = 1$ for all $a \in A$; that the self-loop on g has a zero consumption; and that the agent can observe reaching a goal, i.e., no goal state is lookalike with a non-goal state. This captures the fact that reaching a goal ends the agent's interaction with the environment.

We study the problem of reaching a goal in a most efficient way while preventing resource exhaustion. To this end, we augment the CoPOMDP with a *cost* function $cost\colon S \times A \to \mathbb{R}_{\geq 0}$, stipulating that the self-loops on goal states have zero costs (we can also allow negative costs for some state-action pairs, as long as such pairs can appear only finitely often on each trajectory: these can be used, e.g., as one-time "rewards" for the agent reaching a goal state). Then the *total cost* of a policy π is the quantity

$$TC(\pi) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} cost(S_t, A_t) \right].$$

Resource-Constrained Goal Optimization. A policy π is *safe* if it ensures that the resource is never exhausted; due to the discrete nature of CoPOMDPs, this is equivalent to requiring that the exhaustion probability is zero:

$$\mathbb{P}^\pi(L_t = \bot \text{ for some } t \in \mathbb{N}) = 0.$$

In the *resource-constrained goal optimization (RSGO)* problem, we aim to find a policy π minimizing $TC(\pi)$ subject to the constrain that π is a *safe goal* policy.

Belief Supports for CoPOMDPs. When working with POMDPs, one often does not work directly with histories but with their suitable *statistics*. The *constraints* in the RSGO problem are *qualitative*, and hence qualitative statistics should be sufficient for satisfying resource safety. This motivates the use of *belief supports*. For each history $h \in$ *Hist*, the belief support of h is the set $Bel(h) \subseteq S$ of all states in which the agent can be with a positive probability after observing history h. The standard formal definition of a belief support (Chatterjee et al. 2016) needs to be extended to incorporate resource levels. This extension is somewhat technical, and we defer it to the full version. Similarly to standard belief supports, it holds that for a history $h = \hat{h}ao\ell$ with prefix $\hat{h}$, given $Bel(\hat{h}), a, o, \ell$, and last resource level of $\hat{h}$, we can compute $Bel(h)$ in quadratic time. Thus, the agent can efficiently update its belief support when making a step.

Given a belief support B, state $s \in B$, and any history $\hat{h}ao\ell$ s.t. $B = Bel(\hat{h})$ and $\mathcal{O}(o|t) > 0$ for some $t \in Succ(s,a)$, we say that the belief support $Bel(\hat{h}ao\ell)$ is the s-successor of B under a. Slightly abusing the notation, we denote by $Succ(B,a,s)$ the set of all possible s-successors of B under a. We also denote $Succ(B,a) = \bigcup_{s \in B} Succ(B,a,s)$. Given B and a (or s), $Succ(B,a)$ and $Succ(B,a,s)$ can be computed in polynomial time.

Shielding for CoPOMDPs

Informally, a shield is an algorithm which, in each step, *disables* some actions (based on the available information) and thus prevents the agent from using them. In this paper, we formalize shields as follows:

Definition 2. *A* shield *is a function $\sigma\colon Hist \times A \to \{1, 0\}$. We say that σ* enables *action $a \in A$ in history h if $\sigma(h, a) = 1$, otherwise σ* disables *a.*

We say that policy π conforms *to σ if it never selects a disabled action, i.e., if $\sigma(h, a) = 0$ implies $\pi(h)(a) = 0$, for all $h \in Hist, a \in A$.*

A shield σ is support-based if $\sigma(h, a) = \sigma(h', a)$ for any action a and histories h, h' s.t. $Bel(h) = Bel(h')$ and $\ell_h = \ell_{h'}$. We treat support-based shields as objects of type $2^S \times [ca] \times A \to \{1, 0\}$. A support-based shield σ is succinct *if for every $B \in 2^S$ and every $a \in A$ there is a threshold $\tau_{B,a}$ (possibly equal to ∞) such that $\sigma(B, \ell, a) = 1 \Leftrightarrow \ell \geq \tau_{B,a}$.*

Succinct shield can be represented by a table of size $\mathcal{O}(2^{|S|} \cdot |A|)$ storing the values $\tau_{B,a}$. Hence, we treat them as functions of the type $2^S \times A \to [ca] \cup \{\infty\}$.

RSGO Problem and Shields. We aim to construct shields taking care of the qualitative constraints in the RSGO problem. To this end, our shields need to prevent two types of events: directly exhausting the resource, and getting into a situation where a goal state cannot be reached almost surely without risking resource exhaustion. The latter condition can be formalized via the notion of a trap:

Definition 3. *A tuple (B, ℓ), where $B \subseteq 2^S$, is a* trap *if setting the initial distribution λ_0 to the uniform distribution over B and the initial resource level L_0 to ℓ yields a CoPOMDP in which no safe goal policy exists.*

Moreover, an ideal shield should not over-restrict the agent, i.e., it should allow any behavior that does not produce some of the two events above. The following definition summarizes our requirement for shields.

Definition 4. *A shield σ is* exact *if it has the following three properties:*

- *every policy conforming to σ is safe; and*
- *each policy π conforming to σ avoids traps, i.e., satisfies*
  $\mathbb{P}^\pi((Bel(H_t), L_t)$ *is a trap for some* $t \in \mathbb{N}) = 0$*; and*
- every *safe goal policy conforms to σ.*

Later in the paper, we show how to employ a heuristic search augmented by an exact shield to solve a finite-horizon approximation of the RSGO problem. Before that, we present an algorithm which, given a CoPOMDP $\mathcal{C}$, computes an exact shield for $\mathcal{C}$ and checks whether the RSGO problem for $\mathcal{C}$ has a feasible solution.

Computing Exact Shields

We construct exact shields by computing *threshold levels.* These indicate, for each situation, what is the minimal resource level with which the agent can still manage to almost-surely reach a goal state without exhausting the resource.

Definition 5. *Let $\mathcal{C}$ be a CoPOMDP and $h \in Hist$ be its history. The* threshold level *and* positive-threshold level *of h in $\mathcal{C}$ are the quantities $TLev^{\mathcal{C}}_{=1}(h)$ and $TLev^{\mathcal{C}}_{>0}(h)$, respectively, equal to the smallest resource level $\ell \in [ca]$ that has the following property: if we set the initial distribution λ_0 of $\mathcal{C}$ to the uniform distribution over the set $Bel(h)$ and the initial resource level L_0 to ℓ, then the resulting CoPOMDP has a:*

- *safe goal policy in the case of $TLev^{\mathcal{C}}_{=1}(h)$;*
- *safe positive-goal policy in the case of $TLev^{\mathcal{C}}_{>0}(h)$.*

In the case such a resource level does not exist at all, we set the respective value to ∞. We omit the superscript if $\mathcal{C}$ is clear from the context.

The next lemma connects threshold levels to shields.[1]

Lemma 1. *Let σ be an exact shield. Then for each $h \in Hist$ and each $a \in A$ we have that $\sigma(h, a) = 1$ if and only if ℓ_h is greater than or equal to the smallest number τ s.t. for any $s \in Bel(h)$ and any valid history of the form $hao\ell$ s.t. $\mathcal{O}(o|t) > 0$ for some $t \in Succ(s, a)$, it holds that $\Delta(\tau, s, a) \geq TLev_{=1}(hao\ell)$.*

I.e., an action can be enabled iff all possible outcomes of that action lead to a situation where the current resource level is at least the threshold level for that situation. Note that Lemma 1 entails the existence of a unique (up to values in histories that already exhausted the resource) exact shield. Moreover, since threshold levels of a history h only depend on $Bel(h)$, this exact shield is support-based and succinct.

Computing Threshold Levels in POMDPs

We build on the fact, established in the previous work, that threshold levels can be efficiently computed for *perfectly observable* consumption Markov decision processes, or CoMDPs. In CoMDPs, the agent can perfectly observe the current state S_t and the history of past states. For notational convenience, we treat CoMDPs as a special case of CoPOMDPs in which $\mathcal{Z} = S$ and each state always produces itself as its unique observation: we thus omit the observations and observation function from the description of a CoMDP.

Theorem 1 ((Blahoudek et al. 2020))**.** *In a CoMDP, the values $TLev_{=1}(h)$ and $TLev_{>0}(h)$ depend only on the last observation (i.e., state) of h. Moreover, for each state s, the values $TLev_{=1}(s)$ and $TLev_{>0}(s)$ – the common values of $TLev_{=1}(h)$ and $TLev_{>0}(h)$, respectively, for all h's whose last state equals to s – can be computed in polynomial time.*

The main idea of our approach is to turn a given CoPOMDP P into a *finite* CoMDP that captures the aspects of CoPOMDP dynamics pertaining to threshold levels. Below, we present a construction of such a CoMDP.

Consumption Consistency. The construction assumes that the input CoPOMDP is *consistent*, i.e., $C(s, a) = C(t, a)$ for each pair of lookalike states s, t. Any CoPOMDP can be easily transformed into a consistent one by splitting each probabilistic transition $\delta(s, a)(t)$ with a dummy state $t_{s,a}$ in which the consumption depending on s and a takes place. (The state $t_{s,a}$ emits the value $C(s, a)$ as its observation, which only gives the agent information that he is guaranteed to get in the next step anyway).

Lemma 2. *Given a CoPOMDP $\mathcal{C}$, one can construct, in time linear in the size of $\mathcal{C}$, an equivalent (in terms of policies and their costs) consistent CoPOMDP $\mathcal{C}'$.*

[1] Full proofs, missing constructions, and benchmark details in the full version (Ajdarów, Brlej, and Novotný 2022).

Token CoMDPs. A *token CoMDP* is our generalization of a "POMDP to MDP" construction used in (Baier, Bertrand, and Größer 2008) to prove decidability of almost-sure reachability in (standard) POMDPs. States of the token CoMDP correspond to tuples (B, α), where B is a belief support in the original CoPOMDP and $\alpha \in B$ is a "token", signifying the agent's guess of the current state (α can also equal a special symbol ε, representing an invalid guess). Formally, given a consistent CoPOMDP $\mathcal{C} = (S, A, \delta, \mathcal{Z}, \mathcal{O}, C, R, ca)$ we construct a *token* CoMDP $\mathcal{C}_T = (S_T, A, \delta_T, C_T, R_T, ca)$ such that:

- S_T contains all reachable tuples (B, α) s.t. $B \subseteq 2^S$ is a belief support in $\mathcal{C}$ and α is either an element of B or a special symbol ε ("empty guess");
- for each $(B, \alpha) \in S_T$ and each $a \in A$ we have that $\delta_T((B, \alpha), a)$ is a uniform distribution over the set $Succ_T((B, \alpha), a)$ defined as follows:
  - if $\alpha \neq \varepsilon$, we add to $Succ_T((B, \alpha), a)$ all tuples of the form (B', α'), where $B' \in Succ(B, a)$ and α' satisfies one of the following: either $\alpha' \in Succ(\alpha, a) \cap B'$ or $Succ(\alpha, a) \cap B' = \emptyset$ and $\alpha' = \varepsilon$;
  - if $\alpha = \varepsilon$, we add to $Succ_T((B, \alpha), a)$ all tuples of the form (B', ε), where $B' \in Succ(B, a)$
- for each $(B, \alpha) \in S_T$ and each $a \in A$ we put $C_T((B, \alpha), a) = C(s, a)$ where s is an arbitrary element of B (this definition is correct since $\mathcal{C}$ is consistent).
- R_T contains those tuples $(B, \alpha) \in S_T$ such that $B \subseteq R$.

Pruning Token CoMDPs and Computing Exact Shields. It is *not correct* to directly apply the algorithm of (Blahoudek et al. 2020) to the token MDP, as the following example shows.

Example 1. *Consider the CoPOMDP $\mathcal{C}$ pictured in the left part of Figure 1, with the corresponding token CoMDP in the right (the position of the token is given by the hat symbol, e.g., $\{\hat{p}, q\}$ represents the state $(\{p, q\}, p)$). There exists no safe goal policy from s in $\mathcal{C}$, since there is always a chance of getting stuck in t: after one step, the agent cannot know whether it is in p or q and hence whether to choose action a or b. But in the token CoMDP such a policy clearly exists.*

Instead, we iteratively "prune" the token CoMDP $\mathcal{C}_T = (S_T, A, \delta_T, C_T, R_T, ca)$ by iteratively removing the reloading property from all pairs (B, α) that correspond to a trap, until reaching a fixed point. The resulting CoMDP can be used to compute threshold values in the original CoPOMDP and thus also an exact shield for $\mathcal{C}$. The process is summarized in Algorithm 1. Lines 2–10 compute the threshold levels, the remaining lines extract the shield. The latter part uses an "inverse" $\Psi\colon [ca] \times S \times A \to [ca] \cup \{\infty\}$ of the function Δ s.t. $\Psi(\ell', s, a)$ can be interpreted as the minimal amount of resource we need to have in s so that after playing a we have at least ℓ' units. Formally, if $\ell' = \bot$, also $\Psi(\ell', s, a) = \bot$, irrespective of s, a. Otherwise,

$$\Psi(\ell', s, a) = \begin{cases} \ell' + C(s, a) & \ell' \leq ca - C(s, a) \wedge s \notin R, \\ 0 & \ell' \leq ca - C(s, a) \wedge s \in R, \\ \infty & \ell' > ca - C(s, a). \end{cases}$$

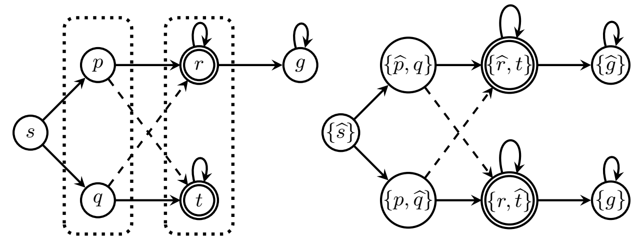

Figure 1: A CoPOMDP $\mathcal{C}$ (left) and its corresponding token CoMDP (right). States r, t are reloads, and g is a goal. There are two actions, a and b. Solid edges represent transitions under action a (with uniform branching in r), and dashed edges represent action b (if both actions behave in the same way in a given state, only the a-edges are pictured). States enclosed in dotted rectangles are indistinguishable, i.e., always emit the same common observation. We omit states $\{s\}$, $\{p, q\}$ and $\{r, t\}$, as these are unreachable from $\{\hat{s}\}$.

Theorem 2. *After the repeat cycle in Algorithm 1 finishes, for any $B \subseteq 2^S$ and $\alpha, \beta \in B$ it holds $TLev^{\mathcal{C}_T}_{>0}(B, \alpha) = TLev^{\mathcal{C}_T}_{>0}(B, \beta)$. Moreover, the computed σ is the unique (support-based succinct) exact shield for $\mathcal{C}$. There is a safe goal policy (i.e., the RSGO problem admits a feasible solution) iff σ enables at least one action for the history of length 0. The algorithm runs in time $\mathcal{O}(2^S \cdot poly(||\mathcal{C}||))$, where $||\mathcal{C}||$ denotes the encoding size of $\mathcal{C}$ (with all integers encoded in binary).*

RSGO Problem and Shielded POMCP

We tackle the RSGO problem by augmenting the POMCP algorithm with our shields.

POMCP. POMCP (Silver and Veness 2010) is a well-known online planning algorithm for POMDPs, based on the *Monte Carlo tree search (MCTS)* paradigm. To select an optimal action, POMCP iteratively searches through and expands the *history tree* of a POMDP, whose nodes correspond to histories. Each iteration consists of a top-down traversal of the explored part of the tree, selecting simulated actions according to the UCT formula (Kocsis and Szepesvári 2006), which balances exploitation with exploration. Once the simulation reaches a yet unexplored node, a *rollout* policy (typically selecting actions at random) is used to further extend the sampled trajectory. The trajectory is then evaluated, and the outcome is back-propagated to adjust action-value estimates along the sampled branch. After multiple such iterations, POMCP selects action with minimal value estimate in the root to be played by the agent. After receiving the next observation, the corresponding child of the root becomes the new root, and the process repeats until a decision horizon is reached.

Search Tree for Shielding. We augment POMCP's tree data structure so that each node additionally contains the information about the current resource level and the current belief support (which is computable for each node using the belief support of the parent.)

Algorithm 1: Computing exact shields.

Input: consistent CoPOMDP $\mathcal{C}$
Output: succinct support-based exact shield σ for $\mathcal{C}$
compute the token CoMDP $\mathcal{C}_T = (S_T, A, \delta_T, C_T, R_T, ca)$
repeat
  $Rem \leftarrow \emptyset$;
  compute $TLev_{>0}^{\mathcal{C}_T}$; /* Theorem 1 */
  foreach $(B, \alpha) \in R_T$ **do**
    if $TLev_{>0}^{\mathcal{C}_T}(B, \alpha) = \infty$ **then**
      foreach $\beta \in B \cup \{\varepsilon\}$ **do**
        $Rem \leftarrow Rem \cup \{(B, \beta)\}$
  $\mathcal{C}_T \leftarrow (S_T, A, \delta_T, C_T, R_T \setminus Rem, ca)$
until $Rem = \emptyset$;
foreach $B \subseteq 2^S, a \in A$ **do**
  $MAX \leftarrow -\infty$;
  foreach $s \in B$ **do**
    $SMAX \leftarrow -\infty$;
    foreach $B' \in Succ(B, a, s)$ **do**
      $\alpha' \leftarrow$ any elem. of B';
      $SMAX \leftarrow \max(SMAX, TLev_{>0}^{\mathcal{C}_T}(B', \alpha'))$;
    $MAX \leftarrow \max(MAX, \Psi(SMAX, s, a))$
  $\sigma(B, a) \leftarrow MAX$
return σ

FiPOMDP. Combining a (support-based exact) shield σ with POMCP yields an algorithm which we call FiPOMDP ("Fuel in POMDPs"). FiPOMDP operates just like POMCP, with one crucial difference: whenever POMCP is supposed to select an action in a node representing history h, FiPOMDP chooses only among actions a such that $\sigma(B(h), a) = 1$. This applies to the simulation/tree update phase (where it selects action optimizing the UCT value among all allowed actions), rollouts, and final action selection (where it chooses the allowed action with minimal value estimate). Since POMCP is an online algorithm operating over a finite decision horizon, FiPOMDP solves the RSGO problem in the following approximate sense:

Theorem 3. *Let N be the decision horizon and consider a finite horizon approximation of the RSGO problem where the costs are accumulated only over the first N steps. Consider any decision step of FiPOMDP and let h be the history represented by the current root node of the search tree. Let p_h be the probability that the action selected by POMCP to be played by the agent is an action used in h by an optimal finite-horizon safe goal policy, and sim the number of simulations used by FiPOMDP. Then for $sim \to \infty$ we have that $p_h \to 1$.*

Shielding Other Algorithms. The shields are algorithm-agnostic, and their usage is not limited to POMCP or MCTS algorithms. Indeed, one of the advantages of our approach is that shields can be used with any algorithm that tracks the current resource level and the current belief support.

Experiments

We implemented FiPOMDP in Python. The algorithm for exact shield computation was implemented on top of the planning algorithm for CoMDPs. (Blahoudek et al. 2020). We wrote our own implementation of POMCP, including the particle filter used for belief approximation (Silver and Veness 2010). The up-to-date link to our implementation can be found in the full version of the paper (Ajdarów, Brlej, and Novotný 2022).[2]

Benchmarks. We evaluated FiPOMDP on three sets of benchmarks. The first benchmark is a toy *resource-constrained Tiger,* a modification of the classical benchmark for POMDPs (Kaelbling, Littman, and Cassandra 1998) adapted from (Brázdil et al. 2016). The goal states represent the situation where the agent has made a guess about the tiger's whereabouts. In the resource-constrained variant, the agent's listening actions consume energy, necessitating regular reloads. During each reload, there is a probability that the tiger switches its position. There is a cost of 10 per each step. Opening the door with tiger/treasure yields cost 5000/-500. We consider two versions: *simple*, where the probability of the observed position of the tiger being correct is 0.85, and *fuzzy*, where this probability is decreased to 0.6.

The second benchmark is a partially observable extension of the *unmanned underwater vehicle (UUV)* benchmark from (Blahoudek et al. 2021). Here, the agent operates in a grid-world, with actions performing movements in the cardinal directions. Movement is subject to stochastic perturbations: the UUV might drift sideways from the chosen direction due to ocean currents. The position sensor is noisy: when the agent visits some cell of the grid, the observed position is sampled randomly from cells in the von Neumann neighborhood of the true cell. We consider 4 gridworld sizes ranging from 8x8 to 20x20. There is a cost of 1 per step, hitting a goal yields "cost" -1000.

The final benchmark, adapted from (Blahoudek et al. 2020), consists of a routing problem for an autonomous electric vehicle *(AEV)* in the middle of Manhattan, from 42nd to 116th Street. Road intersections act as states. At each intersection, the AEV picks a direction to continue (subject to real-world one-way restrictions). It will deterministically move in the selected direction, but the energy consumption is stochastic due to the fluctuations in road congestion. There are three possible consumption levels per road segment, their probability and magnitudes derived from real-world traffic data. (Uber Movement 2019; Straubel 2008). Similarly, the reload states correspond to the real-world positions of charging stations (United States Department of Energy 2019). To add partial observability, we make the consumption probabilities dependent on the unknown *traffic state* (low/medium/peak), which evolves according to a known three-state Markov chain. The cost is equal to the amount of resource consumed in a given step, with a "cost" -1000 when a goal is hit.

[2]Repository address at time of submission: https://github.com/xbrlej/FiPOMDP.

	# States	# Obs	Survival %	Hit %	Avg. cost	Avg. time p. dec. (s)	Shield time (s)
FiPOMDP (shielded)							
Tiger simple	8	6	100	99.5	310.61 ± 1942.14	0.05 ± 0.02	< 1
Tiger fuzzy	8	6	100	57.3	1019.95 ± 2031.03	0.08 ± 0.03	< 1
UUV grid 8x8	64	64	100	98	-969.81 ± 153.61	3.21 ± 1.44	10.65
UUV grid 12x12	144	144	100	92	-898.48 ± 295.92	10.48 ± 3.59	76.99
UUV grid 16x16	256	256	100	87	-839.29 ± 364.92	22.21 ± 5.52	215.18
UUV grid 20x20	400	400	100	87	-839.08 ± 364.83	34.06 ± 8.87	493.87
Manhattan AEV	22434	7478	100	50	2745.3 ± 2845.06	13.02 ± 2.44	65
POMCP (unshielded)							
Tiger simple	8	6	99.9	99.4	311.59 ± 1941.99	0.02 ± 0.01	-
Tiger fuzzy	8	6	86.8	48.5	914.87 ± 1859.81	0.02 ± 0.01	-
UUV grid 8x8	64	64	61	60	-555.75 ± 538.11	1.91 ± 0.27	-
UUV grid 12x12	144	144	8	7	23.76 ± 279.29	9.49 ± 1.23	-
UUV grid 16x16	256	256	4	3	67.52 ± 185.62	18.23 ± 1.60	-
UUV grid 20x20	400	400	6	5	45.77 ± 237.57	28.94 ± 2.96	-
Manhattan AEV	22434	7478	99	56	2352.88 ± 2935.04	13.36 ± 2.38	-

Table 1: Results of experiments. The top part shows results for FiPOMDP, the bottom for the POMCP baseline.

Evaluation. The hardware configuration was: CPU: AMD Ryzen 9 3900X (12 cores); RAM: 32GB; Ubuntu 20.04.

FiPOMDP is the first approach to solving the RSGO problem in CoPOMDPs. Hence, as a baseline to compare with, we chose plain (unshielded) POMCP (with the same hyperparameters), to see how the formal safety guarantees of FiPOMDP influence resource safety in practice. POMCP itself does not consider resource levels, which puts it at a disadvantage. To mitigate this, we treated (only in the POMCP experiments) resource exhaustion as entering a "breakdown" sink state, from which the target can never be reached. Hence, runs exhausting the resource were penalized with the same cost as runs which did not reach the goal.

The results are pictured in Table 1, averaged over 100 runs (1000 for the Tiger benchmark). The first two columns show the number of states and observations. The *Survival %* is the percentage of runs in which the agent *did not* run out of the resource. The *Hit %* is the percentage of runs in which the agent hit the target within the decision horizon. The next column shows an average cost incurred by the agent (± the std. deviation). We also present average time per decision step. The final column shows the time needed to compute the shield (including the computation of the token CoMDP).

We highlight the following takeaway messages: (1.) Although computing an exact shield requires formal methods, our algorithm computed a shield within a reasonable time, even for relatively large (from a formal methods point of view) CoPOMDPs (the Manhattan benchmark). (2.) Shielding is essential for resource safety. The unshielded version never achieved 100% resource safety. In contrast, FiPOMDP *never* exhausted the resource, validating its theoretical guarantees. (3.) The *Hit percentage* and *Cost* results show that the shielded POMCP is consistently able to reach the goal. On the other hand, the hit ratios are sometimes not as high as desired. We suspect that this is because our benchmarks are "non-smooth" in the sense that the costs encountered *before* reaching the goal do not provide much information about a *path* towards the goal. This was partially mitigated using *heavy rollouts* (in particular for the gridworld benchmark, where we used non-uniform rollouts with an increased likelihood of the agent repeating the direction chosen in the previous step). (4.) Since unshielded POMCP tends to exhaust the resource, FiPOMDP has (in all but one of the benchmarks) better hit percentage than the unshielded POMCP. In relatively structureless domains, such as the gridworld, shielding seems to help exploring the state space by pruning away parts from which resource exhaustion cannot be prevented. (5.) The Manhattan benchmark stands out in that the unshielded version performs better in terms of "Hit %" than the shielded one. Still, the unshielded version still is not 100% safe. The benchmark admits a policy for quickly reaching the goal, which carries a small risk of resource exhaustion. The unshielded agent takes this policy, while the shielded agent computes a policy that is resource-safe at the cost of slower progress. This shows that shields protect even against low (though practically significant) exhaustion risks.

Conclusion

We presented a shielding algorithm for consumption POMDPs with resource safety and goal reachability objectives. We combined our shields with the POMCP planning algorithm, yielding a heuristic approach to solving the RSGO problem. An interesting direction for future work is to combine our shielding algorithm with alternative approaches to POMDP planning.

Acknowledgments

This work is supported by the Czech Science Foundation grant No. 21-24711S. We thank all the anonymous reviewers for providing feedback on the preliminary versions of this paper.

References

Ajdarów, M.; Brlej, Š.; and Novotný, P. 2022. Shielding in Resource-Constrained Goal POMDPs. arXiv:2211.15349.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. In McIlraith, S. A.; and Weinberger, K. Q., eds., *AAAI Conference on Artificial Intelligence, (AAAI 2018)*, 2669–2678. AAAI Press.

Altman, E. 1999. *Constrained Markov Decision Processes*. Chapman&Hall/CRC. ISBN 9780849303821.

Baier, C.; Bertrand, N.; and Größer, M. 2008. On Decision Problems for Probabilistic Büchi Automata. In Amadio, R., ed., *Foundations of Software Science and Computational Structures (FOSSACS'08)*, 287–301. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-78499-9.

Bhattacharya, S.; Badyal, S.; Wheeler, T.; Gil, S.; and Bertsekas, D. 2020. Reinforcement Learning for POMDP: Partitioned Rollout and Policy Iteration With Application to Autonomous Sequential Repair Problems. *IEEE Robotics and Automation Letters*, 5(3): 3967–3974.

Blahoudek, F.; Brázdil, T.; Novotný, P.; Ornik, M.; Thangeda, P.; and Topcu, U. 2020. Qualitative Controller Synthesis for Consumption Markov Decision Processes. In Lahiri, S. K.; and Wang, C., eds., *Computer Aided Verification (CAV 2020)*, 421–447. Cham: Springer.

Blahoudek, F.; Cubuktepe, M.; Novotný, P.; Ornik, M.; Thangeda, P.; and Topcu, U. 2021. Fuel in Markov Decision Processes (FiMDP): A Practical Approach to Consumption. In Huisman, M.; Păsăreanu, C.; and Zhan, N., eds., *Formal Methods*, 640–656. Cham: Springer International Publishing. ISBN 978-3-030-90870-6.

Bouyer, P.; Fahrenberg, U.; Larsen, K.; Markey, N.; and Srba, J. 2008. Infinite Runs in Weighted Timed Automata with Energy Constraints. In *Proceedings of FORMATS 2008*, volume 5215, 33–47.

Brázdil, T.; Chatterjee, K.; Chmelik, M.; Gupta, A.; and Novotný, P. 2016. Stochastic Shortest Path with Energy Constraints in POMDPs. *CoRR*, abs/1602.07565.

Brázdil, T.; Chatterjee, K.; Kučera, A.; and Novotný, P. 2012. Efficient Controller Synthesis for Consumption Games with Multiple Resource Types. In *Proceedings of CAV 2012*, volume 7358, 23–38.

Brázdil, T.; Klaška, D.; Kučera, A.; and Novotný, P. 2014. Minimizing Running Costs in Consumption Systems. In Biere, A.; and Bloem, R., eds., *Computer Aided Verification 2014*, volume 8559 of *LNCS*, 457–472. Springer International Publishing. ISBN 978-3-319-08866-2.

Brázdil, T.; Kučera, A.; and Novotný, P. 2016. Optimizing the Expected Mean Payoff in Energy Markov Decision Processes. In *14th International Symposium on Automated Technology for Verification and Analysis*, 32–49.

Chakrabarti, A.; de Alfaro, L. d.; Henzinger, T. A.; and Stoelinga, M. 2003. Resource Interfaces. In Alur, R.; and Lee, I., eds., *Proceedings of EMSOFT 2003*, volume 2855 of *LNCS*, 117–133. Heidelberg: Springer.

Chatterjee, K.; Chmelík, M.; Gupta, R.; and Kanodia, A. 2016. Optimal cost almost-sure reachability in POMDPs. *Artificial Intelligence*, 234: 26–48.

Chatterjee, K.; Novotný, P.; Pérez, G. A.; Raskin, J.; and Zikelic, D. 2017. Optimizing Expectation with Guarantees in POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3725–3732. AAAI Press.

Degorre, A.; Doyen, L.; Gentilini, R.; Raskin, J.-F.; and Toruńczyk, S. 2010. Energy and mean-payoff games with imperfect information. In *24th International Workshop on Computer Science Logic*, 260–274.

Dräger, K.; Forejt, V.; Kwiatkowska, M. Z.; Parker, D.; and Ujma, M. 2015. Permissive Controller Synthesis for Probabilistic Systems. *Log. Methods Comput. Sci.*, 11(2).

Eaton, C. M.; Krakow, L. W.; Chong, E. K.; and Maciejewski, A. A. 2018. Fuel Efficient Moving Target Tracking Using POMDP with Limited FOV Sensor. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, 331–336.

García, J.; and Fernández, F. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16(42): 1437–1480.

Junges, S.; Jansen, N.; Dehnert, C.; Topcu, U.; and Katoen, J. 2016. Safety-Constrained Reinforcement Learning for MDPs. In Chechik, M.; and Raskin, J., eds., *Proceedings of TACAS 2016.*, volume 9636 of *Lecture Notes in Computer Science*, 130–146. Springer.

Junges, S.; Jansen, N.; and Seshia, S. A. 2021. Enforcing Almost-Sure Reachability in POMDPs. In Silva, A.; and Leino, K. R. M., eds., *Computer Aided Verification*, 602–625. Cham: Springer International Publishing. ISBN 978-3-030-81688-9.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1): 99–134.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In Fürnkranz, J.; Scheffer, T.; and Spiliopoulou, M., eds., *European Conference on Machine Learning (ECML 2006)*, volume 4212 of *LNCS*, 282–293. Springer.

Mayr, R.; Schewe, S.; Totzke, P.; and Wojtczak, D. 2017. MDPs with energy-parity objectives. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, 1–12.

Mitchell, D.; Corah, M.; Chakraborty, N.; Sycara, K.; and Michael, N. 2015. Multi-robot long-term persistent coverage with fuel constrained robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 1093–1099.

Notomista, G.; Ruf, S. F.; and Egerstedt, M. 2018. Persistification of Robotic Tasks Using Control Barrier Functions. *IEEE Robotics and Automation Letters*, 3(2): 758–763.

Poupart, P.; Malhotra, A.; Pei, P.; Kim, K.; Goh, B.; and Bowling, M. 2015. Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes. In *AAAI 2015.*, 3342–3348. AAAI Press.

Silver, D.; and Veness, J. 2010. Monte-Carlo Planning in Large POMDPs. In *Neural Information Processing (NIPS 23)*, 2164–2172. Curran Associates, Inc.

Straubel, J. B. 2008. Roadster Efficiency and Range. https://www.tesla.com/blog/roadster-efficiency-and-range. Accessed: 2023-03-23.

Uber Movement. 2019. Traffic Speed Data for New York City. https://movement.uber.com/. Accessed: 2023-03-23.

Undurti, A.; and How, J. P. 2010. An Online Algorithm for Constrained POMDPs. In *International Conference on Robotics and Automation (ICRA'17)*, 3966–3973. IEEE.

United States Department of Energy. 2019. Alternative Fuels Data Center. https://afdc.energy.gov/stations/.