Safe Policies for Factored Partially Observable Stochastic Games

Carr, Steven; Jansen, Nils; Bharadwaj, Suda; Spaan, M.T.J.; Topcu, Ufuk

DOI
10.15607/RSS.2021.XVII.079

Publication date
2021

Document Version
Final published version

Published in
Robotics: Science and System XVII

Citation (APA)
Carr, S., Jansen, N., Bharadwaj, S., Spaan, M. T. J., & Topcu, U. (2021). Safe Policies for Factored Partially Observable Stochastic Games. In D. A. Shell, M. Toussaint, & M. A. Hsieh (Eds.), *Robotics: Science and System XVII* https://doi.org/10.15607/RSS.2021.XVII.079

Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Safe Policies for Factored Partially Observable Stochastic Games

Steven Carr[1], Nils Jansen[2], Suda Bharadwaj[1], Matthijs T. J. Spaan[3] and Ufuk Topcu[1]
[1] The University of Texas at Austin, Texas, USA
[2] Radboud University, Nijmegen, The Netherlands
[3] Delft University of Technology, The Netherlands
Email: {stevencarr,sudab,utopcu}@utexas.edu, n.jansen@science.ru.nl, m.t.j.spaan@tudelft.nl

Abstract—**We study planning problems where a controllable agent operates under partial observability and interacts with an uncontrollable opponent, also referred to as the adversary. The agent has two distinct objectives: To maximize an expected value and to adhere to a safety specification. Multi-objective partially observable stochastic games (POSGs) formally model such problems. Yet, even for a single objective, the task of computing suitable policies for POSGs is theoretically hard and computationally intractable in practice. Using a factored state-space representation, we define a decoupling scheme for the POSG state space that—under certain assumptions on the observability and the reward structure—separates the state components relevant for the reward from those relevant for safety. This decoupling affects the possibility to compute provably safe and reward-optimal policies in a tractable two-stage approach. In particular, on the fully observable components related to safety, we exactly compute the set of policies that captures all possible safe choices against the opponent. We restrict the agent's behavior to these safe policies and project the POSG to a partially observable Markov decision process (POMDP). Any reward-maximal policy for the POMDP is then guaranteed to be safe and reward-maximal for the POSG. We showcase our approach's feasibility using high-fidelity simulations of two case studies that concern UAV path planning and autonomous driving. Moreover, to demonstrate the practical applicability, we design a physical experiment involving a robot decision making problem under energy constraints that is motivated by a paired helicopter with NASA's Perseverance Mars rover.**

I. Introduction

Systems involving autonomous agents that make decisions in uncertain and adversarial environments under incomplete information can be represented as partially observable stochastic games (POSGs). Due to the generality of POSGs, they cover a large number of application areas such as robotics [25], cybersecurity [17], and air-traffic control [38]. However, computing a reward-optimal policy for an agent in a POSG is notoriously hard or impossible to solve [13]. Existing literature examines mostly approximate methods in small settings while realistic problems remain largely intractable [11, 26, 16].

Problem setting: This work focuses on *one-sided POSGs* where the (controllable) Agent 1 operates under partial observability and has to account for the behavior of the uncontrollable Agent 2, which has full observability of the state space. Within a multiple-objective *one-sided* POSG problem, Agent 1 shall act according to a *performance* and a *safety* objective. The performance objective is to optimize a *cumulative (discounted) reward* or cost measure, take as an example an agent that needs to perform delivery tasks with a minimal expected number of steps. As (required) safety objectives we use *almost-sure specifications*, where the probability to reach a set of *avoid* states needs to be zero. Such specifications restrict Agent 1's behavior with respect to that of Agent 2. For example, Agent 1 may have to remain within a certain distance to Agent 2 at all times. The goal is for Agent 1 to strictly maintain safety while optimizing the expected reward as much as possible. For fully observable MDPs, such multi-objective problems can be solved efficiently [7, 43]. However, directly computing a safe POSG policy is EXPTIME-complete [5], and this does not even account for the (undecidable) task of finding the optimal reward policy.

Our approach: The goal of our approach is to employ domain knowledge in order to solve the generally intractable problem. More precisely, we demonstrate that for many natural examples, there is an inherent separability of performance and safety. To provide a principled way to facilitate such a separation of concerns, we employ a factored state-space representation [45, 42] to identify relevant features for the individual objectives. We define a *reward separability* and a *safety separability* assumption based on these features. Intuitively, for the case where Agent 2's behavior is not relevant for the expected reward objective, we design a formal projection from the POSG to a partially observable Markov decision process (POMDP) with a reduced state space that neglects Agent 2. Likewise, for the safety objective, if Agent 1 can fully observe the features that relate to Agent 2, we perform a projection that results in a 2-player stochastic game with full observability [4].

These projections allow the use of efficient and mature tools. Let us recall that the safety objective is critical and only rigorous methods are acceptable. To generate policies that are safe against *all* possible behaviors of Agent 2, we compute a so-called *maximally permissive policy* that allows multiple safe actions at each state. Intuitively, this permissive policy contains all policies for Agent 1 that satisfy the safety specification. We can reduce this task to a tractable mixed-integer linear program (MILP) [10, 12].

After computing the exact set of safe policies, we exclude potential actions that are not compliant with the permissive (safety-ensuring) policy from the POSG. The restricted model is then projected to a POMDP using the aforementioned reward separability. While computing an optimal infinite-horizon POMDP policy is undecidable in general [28], tractable approaches that utilize approximate [14, 8], point-based [33, 41], or Monte-Carlo-based [39] methods can compute near-optimal solutions for environments with millions of states. We show and exploit that any (reward-optimal) policy for the POMDP is reward-optimal and safe for Agent 1 in the POSG.

Contribution: Our main contribution is a principled scheme to solve POSG scenarios with multiple objectives related to safety and reward optimization. The aforementioned separability assumptions are, while strong in general, motivated by realistic examples and justified by the fact that to the best of our knowledge there is no method that is able to solve the relevant problem we address in this paper.

Case studies: We demonstrate the authenticity of the assumptions through several new POSG models that capture real-world examples via high-fidelity simulations, and provide a physical case study that further demonstrates the relevance of our approach. In particular, our scenarios concern a UAV delivery task, a Mars-rover-inspired charging assurance problem, and autonomous vehicles operating around pedestrians. In each example, we utilize explicit domain knowledge to separate the setting according to the features.

A. Tethered UAV Delivery: A Motivating Example

Motivated by an Amazon concept for a balloon-based airborne fulfillment center [2], we consider the following case study where a controllable UAV (Agent 1) and an uncontrollable balloon-based dispatcher (Agent 2) operate in an unknown environment (see Fig. 1). A company wishes to send materials from its headquarters to a building site on the other side of town. The UAV agent attempts to collect materials by landing in a given location and then deliver them to another location (A_1 and then A_2 in Fig. 1) in a cost-minimal manner for distance and fuel consumption. Once landed at A_1, the dispatching balloon recalls the UAV and relaunches it randomly back into the environment to deliver the material to its destination. Furthermore, in a dense urban environment, Agent 1 itself cannot perform localization, only inferring its local position.

Features: Only the UAV's physical location in the environment is relevant to the UAV agent's accumulated reward. The actions taken by the dispatcher have no impact on this reward. Additionally, only the UAV's location in relation to the dispatcher is important to the safety specification, shown by the relevant communication window W in Fig. 1.

B. Related Work

Closest to this work are factored approaches to POMDPs [22] and specifically *mixed-observability MDPs* separate observable and hidden state features [32]. One can apply similar separation assumptions for the case when a set of agents cooperating under partial observablility in a decentralized POMDP [29, 30].

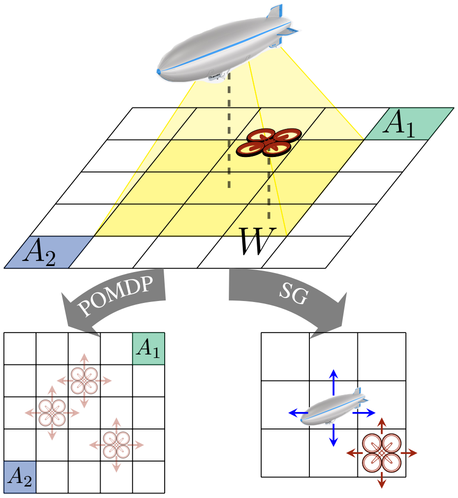


Fig. 1: Example environment with a controlled UAV (Agent 1) and a dispatcher (Agent 2). Agent 1 needs to stay within the observable window (yellow) and visit two target locations (green headquarters and purple building site). Agent 1 can (only) observe these two landmarks when flying overhead (via distinct observations A_1 and A_2), and the relative position of the dispatcher within the observable window.

The complexity of finding policies that satisfy multiple objectives in POSGs has been explored in [5]. The reference [6] further demonstrates the utility of randomized policies and the undecidability of one-sided POSGs. Direct methods such as dynamic programming are intractable in practice [13]. We list the following approaches that can compute policies on small POSGs: using a memory bounded representation of the value function [11]; heuristically reformulating as a Bayesian game [26]; performing value iteration and applying belief-based heuristics [16]; and leveraging marginal probabilities to compactly represent the belief spaces [18] . While these general approaches rely on fewer assumptions than the proposed, dedicated, method, they are only tractable for model sizes of less than 10^6 states.

Additionally, similar works that simplify the computation under environment assumptions rely on some notion of a *public observation* to reduce or collapse the belief space for Agent 2[15, 24, 30]. Other approaches reduce POSGs to tractable POMDPs using value function approximators [19]. However, none of the previous deal with multiple objectives or provide guarantees that the policy is safe with respect to a safety objective.

A conceptually similar approach involves applying a safe permissive policy to restrict the action choices available for performing reinforcement learning to find a safe and optimal policy for a fully observable MDP [21]. The concept of guaranteeing safety at runtime is referred to as *shielding* [23]. Recent works employ shields in reinforcement learning [1], for Markov decision processes (MDP) [20], or for multi-agent systems [3], yet, none involve partially observable systems.

II. Partially Observable Games

We recall probabilistic models, policies, and specifications.

A. Models

Sequential decision making problems under state uncertainty and interaction with adversarial behavior are captured by partially observable stochastic games (POSGs). We consider turn-based games where Agent 1 has partial, and the adversarial agent (Agent 2) has full observability [16].

Let $\mu\colon X \to [0,\, 1] \subseteq \mathbb{R}$ denote a *probability distribution* over X with $\sum_{x \in X} \mu(x) = \mu(X) = 1$. The set of all distributions on X is $\mathit{Distr}(X)$. The support of a distribution is $\mathrm{supp}(\mu) = \{x \in X \mid \mu(x) > 0\}$.

Formally, a *two-player one-sided POSG* is a tuple $\mathcal{G} = (S, I, \mathit{Act}, O, Z, \mathcal{P}, \mathcal{R})$ where $S = S_1 \cup S_2$ is a finite state space for Agent 1 and Agent 2, respectively. I is the initial distribution over states of Agent 1 that gives the probability $I(s)$ to start in state $s \in S_1$, and $\mathit{Act} = \mathit{Act}_1 \cup \mathit{Act}_2$ is a finite space of joint actions for both agents. Z is a finite observation space and $O\colon S_1 \to \mathit{Distr}(Z)$ is the observation model for Agent 1 that returns the probability $O(z|s)$ of observing $z \in Z$ in state $s \in S_1$. $\mathcal{P}(s'|s,a)$ is a transition model that represents the conditional probability of moving to a successor state $s' \in S$ after executing action $a \in A$ in state $s \in S$. Finally when executing an action $a \in \mathit{Act}_1$ in state $s \in S_1$, Agent 1 receives a scalar reward $\mathcal{R}(s,a)$. Note that rewards for Agent 2 are not relevant in our setting, as we model this agent as an adversary for safety concerns.

The game starts at the initial state $s \in S_1$ that has been randomly selected via I. Agent 1 cannot observe s directly but receives observation $z \in Z$ with probability $O(z|s)$. Upon that observation, Agent 1 picks an action $a_1 \in \mathit{Act}_1$ and receives the reward $\mathcal{R}(s, a_1)$. The next state will be $s' \in S_2$ with probability $\mathcal{P}(s'|s, a_1)$ and it is Agent 2's choice. A path through a POSG $\mathcal{G}$ is a sequence π of states and actions. The observation function O applied to π yields an *observation sequence* $O(\pi)$ of observations and actions.

B. Policies and properties

The notion of a *belief* for Agent 1 describes the probability of being in certain state based on the current observation sequence $O(\pi)$. Formally, a belief b is a distribution $b \in \mathit{Distr}(S_1)$ over the states of Agent 1. A policy σ for a one-sided POSG $\mathcal{G}$ is the tuple $\sigma = (\sigma_1, \sigma_2)$ where σ_i is a policy for Agent i. The (stochastic) policy σ_1 for Agent 1 maps a *belief* b to a distribution over actions, $\sigma_1\colon \mathit{Distr}(S_1) \to \mathit{Distr}(\mathit{Act})$. Agent 1 needs to account for all possible choices for Agent 2 in an adversarial manner. Therefore, there is no need to specify Agent 1's policies further.

Agent 1 aims to find a policy σ_1 that maximizes the expected discounted reward $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t\right]$, where γ^t with $0 \leq \gamma^t \leq 1$ is the discount factor and $\mathcal{R}_t$ is the reward Agent 1 receives at time t. Further, we can characterize a policy σ by a value function V^σ which is defined via the expected discounted reward. Note that in general, the expected discounted reward also depends on Agent 2's policy σ_2. Moreover, we consider a *safety objective* that necessitates to avoid certain bad states from $B \subseteq S$ *almost-surely*, that is, with probability one. We denote such an objective in the style of temporal logic constraints [34] by $\varphi = \neg\Diamond B$. A policy σ_1 for Agent 1 *robustly satisfies* the safety objective φ if B is never reached under σ_1 for all policies σ_2 for Agent 2.

If $S_2 = \emptyset$ and $s_{\mathrm{I}} \in S_1$, the POSG is a *partially observable Markov decision process* (POMDP). A policy for a POMDP is then a policy for Agent 1. If the states S_1 of Agent 1 are fully observable, the POSG is a mere *(2-player) stochastic game*. This game can be efficiently solved using linear programming, policy iteration, or value iteration.

III. Approach

In this section, we first state and discuss the formal problem of this paper, and then we introduce the overall approach.

A. Problem Statement

> We are given a one-sided POSG $\mathcal{G}$ and a safety specification $\varphi = \neg\Diamond B$. The problem is to determine a policy for Agent 1 that achieves the maximal expected reward under all policies that robustly satisfy the safety specification.

Recall that this multi-objective problem is, in general, undecidable [37]. To obtain a tractable solution for the problem, we define several assumptions on the behavior of both agents based on concrete domain knowledge. These assumptions are motivated by concrete use cases as, for instance, the UAV example as described in the introduction.

B. Factored POSGs

We first define a factored state representation for POSGs and utilize the direct access to the features of a POSG setting.

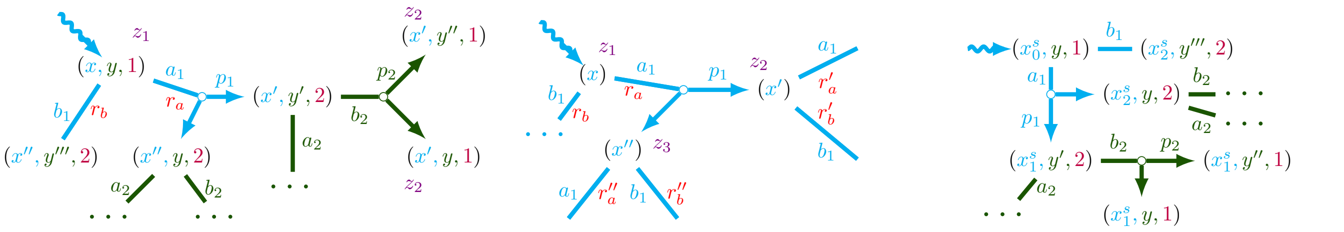
A state $s \in S$ of the POSG is built by a two-dimensional feature vector $s = (x, y) \in X \times Y$ with finite domains $\mathrm{dom}(x) = X$ and $\mathrm{dom}(y) = Y$. For the sake of readability, we start by using only these two features. To achieve more fine-grained representations, the variables may be built by the cross product of further variables, that is, $X = X_1 \times \ldots \times X_n$ and $Y = Y_1 \times \ldots \times Y_m$ for $n, m \in \mathbb{N}_{>0}$. As we consider a turn-based game, we additionally assume a turn-variable t with $\mathrm{dom}(t) = \{1, 2\}$. Thereby, we have $S = X \times Y \times \{1, 2\}$. The states of Agent i are

$$S_i = \{(x, y, t) \in X \times Y \times \{1, 2\} \mid t = i\}\ .$$

For example, a state $(x, y, 1)$ belongs to Agent 1.

C. Reward separability assumption

With the factored state space representation in place, we are ready to formulate our first assumption. Let us start with a short motivation. In the aforementioned UAV delivery example, the feature space X represents the *possible physical locations* of the UAV (Agent 1) in the environment. The feature space Y represents the *relative positions* of Agent 1 from the dispatcher (Agent 2). We observe that actions by the dispatcher can only impact the relative position $y \in Y$ between itself and the UAV, but *cannot* directly impact the



(a) Factored POSG (Example 1).

(b) POMDP (Example 2).

(c) Stochastic game (Example 3).

Fig. 2: Example executions for a one-sided factored POSG and its corresponding projected POMDP and stochastic game with a) $p_1 = \mathcal{P}_1(s'|s, a_1)$ and $p_2 = \mathcal{P}_2(y''|y', b_2)$, b) $p_1 = \mathcal{P}^r(x'|x, a_1)$ and c) $p_1 = \mathcal{P}^s(s^s_1|s^s_0, a_1)$ and $p_2 = \mathcal{P}^s(s^s_2|s^s_1, b_2)$.

physical location $x \in X$ of Agent 1. Moreover, rewards are only affected by the physical location of the UAV, and thus independent of actions of the dispatcher.

We transfer such knowledge about the effects of actions into the POSG towards a feature separation.

We refine the transition model $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ by requiring that $\mathcal{P}_2$ is of the form

$$\mathcal{P}_2 \colon Y \times Act_2 \to Distr(Y). \tag{1}$$

Essentially, actions by Agent 2 affect only feature y. We also assume, the initial distribution I is defined over x.
Moreover, the reward and the observation model depend only on the value of the x-feature. That gives us the following functions that we project onto the relevant features.

$$O \colon X \to Distr(Z), \quad \mathcal{R} \colon X \times Act_1 \to \mathbb{R}. \tag{2}$$

The following technical example explains the corresponding evolution of a POSG.

Example 1. *Consider the excerpt of a POSG in Fig. 2a. At state $s = (x, y, 1)$, Agent 1 carries out an action $a_1 \in Act_1$ and reaches the state $s' = (x', y', 2)$ with probability $p_1 = \mathcal{P}_1(s'|s, a_1)$. Recall that Agent 1's actions can change both features. Moreover, at state $(x, y, 1)$, Agent 1 receives observation $z_1 \in Z$ with probability $O(z_1|x)$, and receives reward $r_a = \mathcal{R}(x, a_1)$ upon executing action $a_1 \in Act_1$. From $s' = (x', y', 2)$, Agent 2 picks action $b_2 \in Act_2$ and with probability $p_2 = \mathcal{P}_2(y''|y', b_2)$ reaches state $s'' = (x', y'', 1)$, affecting only the y-feature.*

Projection to a POMDP: We aim to exploit the reward separability assumption towards a separation of concerns. We will construct a projection of the POSG states and the transition model to the x-features, resulting in a POMDP that purely reflects the choices of Agent 1. To that end, we define the *reward feature projection* $f^r \colon S \to X$ with $f^r((x, y, t)) = (x)$ for all states $(x, y, t) \in X \times Y \times \{1, 2\}$. Lifting this projection to distributions, we have $f^r \colon Distr(S) \to Distr(X)$ with $f^r(\mu(x, y, t)) = \mu(x)$ for all $(x, y, t) \in X \times Y \times \{1, 2\}$ and distributions $\mu \in Distr(S)$. We are ready to define a POMDP for Agent 1.

Definition 1 (Projected POMDP). *Given a one-sided POSG $\mathcal{G} = (S, I, Act, O, Z, \mathcal{P}, \mathcal{R})$, a factored representation with x and y features where the reward separability assumption holds, the reward-projected POMDP is given by $\mathcal{M} = (S^r, I^r, Act^r, O^r, Z^r, \mathcal{P}^r, \mathcal{R}^r)$ with*

- $S^r = X$,
- $I^r(x) = f^r(I(s))$ *for all* $x \in S^r$ *with* $f^r(s) = x$,
- $Act^r = Act_1$,
- $Z^r = Z$,
- $O^r(x) = O(s)$ *for all* $x \in S^r$ *with* $f^r(s) = x$,
- $\mathcal{R}^r(x, a_1) = \mathcal{R}(s, a_1)$ *for all* $x \in S^r$ *with* $f^r(s) = x$, $a_1 \in Act^r$,
- $\mathcal{P}^r(x, a) = f^r(\mathcal{P}(x, a))$ *for all* $x \in S^r$ *with* $f^r(s) = x$.

Note that the projection for the initial distribution is unique because it only depends on the value of x, that is, $f^r(I(s)) = f^r(I(s'))$ for $s \neq s'$ but $f^r(s) = f^r(s')$.

Example 2. *Consider the projected POMDP in Fig. 2b. At state s with feature x, Agent 1 has observation $z_1 \in Z$ with probability $O(z_1|x)$, carries out action $a_1 \in Act_1$ and reaches the state s' with feature x' with probability $p_1 = \mathcal{P}^r(x'|x, a_1)$ receiving reward $r_a = \mathcal{R}^r(x, a_1)$.*

The projection allows us to directly relate the expected reward in the POMDP to the original POSG.

Theorem 1 (Reward consistency). *For the POMDP $\mathcal{M}$ and each policy σ, there exists a POSG policy σ_1 for Agent 1 such that the expected reward is the same for both policies.*

Proof sketch: We are given a policy σ_1 for the POMDP that has value/expected reward $V^{\sigma_1}(b_0)$. We construct a POSG policy σ_1 for Agent 1 such that for all beliefs b over POSG states and actions $a_1 \in Act_1$ of Agent 1 we have that

$$\sigma_1(a_1|b) = \sigma_1(a_1|b') \text{ with } b(s) = b'(x) \text{ and } s = (x, y, 1)$$

for belief b' over POMDP states. Recall that b' is only defined over the x-features (Equation 2). We now assume that there exists a policy σ_2 for Agent 2 such that the full POSG policy $\sigma = (\sigma_1, \sigma_2)$ induces an expected reward $V^{\sigma} \neq V^{\sigma_1}$ that is different from the expected reward under σ_1 in the POMDP. However, the reward separability assumption ensures that actions of Agent 2 cannot induce a change in the x-feature (Equation 1) which uniquely defines the reward model (Equation 2), leading to a contradiction. ■

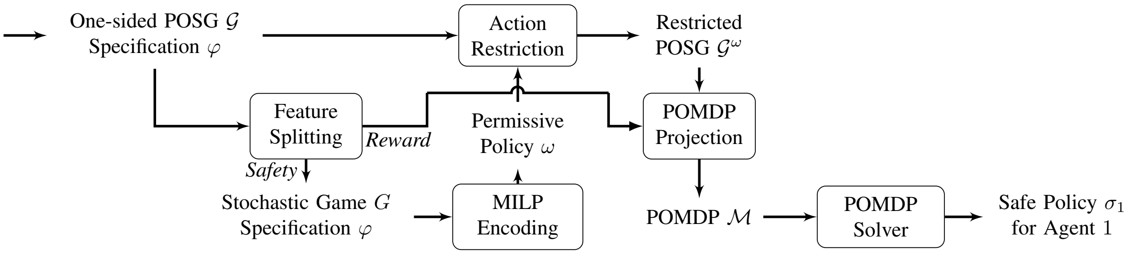


Fig. 3: Process for decomposition and solving the factored POSG.

D. Safety separability assumption

We refer again to the motivating example (Section I-A) where the UAV (Agent 1) follows a safety specification to never collide with the dispatcher (Agent 2), and shall also never move out of visible range (outside of yellow region W in Fig. 1). In this case, Agent 1 only has full observability of feature space Y, which is the relative position to Agent 2 (bottom right of Fig. 1). We leverage this domain knowledge to guide the concept of safety separability.

We assume that the x-feature is separable into features x^r (partially observable) and x^s (fully observable) with domains $\operatorname{dom}(x^r) = X^r$ and $\operatorname{dom}(x^s) = X^s$, which is common in mixed observability MDPs [32]. We call x^r the *reward-relevant* and x^s the *safety-relevant* part of the x-feature. In the context of the motivating example, there is no mixed observability over X and thus $X^r = X$ and $X^s = \emptyset$. We assume now that actions of Agent 1 only affect the x-features x^r and x^s. We have $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ and $\mathcal{P}_1$ is of the form

$$\mathcal{P}_1 \colon X^r \times X^s \times Act_1 \to Distr(X). \tag{3}$$

Additionally, we refine the observation model which now solely depends on x_r, that is $O \colon X^r \to Distr(Z)$. For brevity, we can also assume that the reward model depends only on the x-feature, but that is not strictly necessary. Recall the safety specification $\varphi = \neg\Diamond B$ with $B \subseteq S$. We assume B depends exclusively on the x^s and the y-features, that is, $(x_1^r, x^s, y, t) \in B$ implies that $(x_2^r, x^s, y, t) \in B$ for all $x_1^r, x_2^r \in X^r$, $x^s \in X^s$, $y \in Y$ and $t \in \{1, 2\}$.

Projection to a stochastic game: As the partial observability depends only on the feature x^r, and the safety only on x^s and y, we can now restrict the POSG by ignoring the x^r feature and achieve a fully observable stochastic game. In particular, we define a similar projection as before, with the *safety-feature projection* $f^s \colon S \to X^s \times Y \times \{1, 2\}$ with $f^s((x^r, x^s, y, t)) = (x^s, y, t)$ for all states $(x^r, x^s, y, t) \in X^r \times X^s \times Y \times \{1, 2\}$. Lifting this projection to distributions, we have $f^s \colon Distr(S) \to Distr(X^s \times Y \times \{1, 2\})$ with $f^s(\mu(x^r, x^s, y, t)) = \mu(x^s, y, t)$ for all $(x^r, x^s, y, t) \in X^r \times X^s \times Y \times \{1, 2\}$ and distributions $\mu \in Distr(S)$.

Definition 2 (Projected stochastic game)**.** *Given a one-sided POSG $\mathcal{G} = (S, I, Act, O, Z, \mathcal{P}, \mathcal{R})$, a factored representation with x^r, x^s and y features where the safety separability assumption holds, the safety-projected (fully observable) stochastic game is given by $\hat{\mathcal{G}} = (S^s, I^s, Act^s, \mathcal{P}^s)$ with*

- *$S^s = (X^s, Y, \{1, 2\})$ with $S_i^s = X^s \times Y \times \{i\}$ for $i \in \{1, 2\}$*
- *$I^s(x^s, y, 1) = f^s(I(s))$ for all $(x^s, y, t) \in S^s$ with $f^s(s) = (x^s, y, t)$,*
- *$Act^s = Act_1 \cup Act_2$,*
- *$\mathcal{P}^s(x^s, y, t, a) = f^s(\mathcal{P}(s))$ for all $(x^s, y, t) \in S^s$ with $f^s(s) = (x^s, y, t)$.*

Example 3. *Consider the projected stochastic game in Fig. 2c. At state $s_0 = (x_0^s, y, 1)$, Agent 1 takes action $a_1 \in Act^s$ and reaches the state $s_1^s = (x_1^s, y', 2)$ with probability $p_1 = \mathcal{P}^s(s_1^s | s_0^s, a_1)$. From s_1^s, Agent 2 picks action $b_2 \in Act^s$ and with probability $p_2 = \mathcal{P}^s(s_2^s | s_1^s, b_2)$ state $s_2^s = (x_1^s, y'', 1)$ is reached.*

E. Method

With the reward and safety separability assumptions in place and the dedicated projections, we are now ready to outline our method, see Fig. 3 for the flow. Given a POSG setting, we identify if a feature split as necessary for the assumptions is possible and assign the features accordingly to the state space of the POSG. We first project the POSG $\mathcal{G}$ to a stochastic game $\hat{\mathcal{G}}$ according to Def. 2 and the safety-relevant features. In particular, we leverage the fact that $\hat{\mathcal{G}}$ is fully observable. For such games, a policy that ensures safety for Agent 1 need only be a deterministic (or pure) policy of the form $\sigma_1 \colon S_1^s \to Act_1$.

Now, we compute a so-called *permissive* (deterministic) policy of the form $\omega \colon S \to 2^{Act_1}$. Such permissive policies describe sets of policies for Agent 1 that choose sets of actions from Act_1 at each state $s \in S$. As an example, the permissive policy $\omega(s) = \{a_1, a_2\}$ contains the policies σ_1 and σ_1' with $\sigma_1(s) = a_1$ and $\sigma_1'(s) = a_2$.

We leverage a method based on *mixed-integer linear programming* (MILP) to compute such a permissive policy. The MILP encoding imposes a penalty when an action is removed, as described in [10].

For instance, consider an unsafe region $B = \{y'''\} \in Y$ in Example 1, the permissive policy $\omega(f^s(s)) = \{a_1\}$ at state $s = (x, y, 1)$ restricts the action of b_1 (see Fig. 2a where taking b_1 at $(x, y, 1)$ leads to y''' with probability 1). Under these new restrictions and dropping the assumption that all actions are available at each state, we reformulate the original POSG $\mathcal{G}$ to the restricted POSG $\mathcal{G}^{\omega}$ with respect to the permissive policy whereby for all states $s \in S_1$ the available actions are defined by the function $Act_{\omega}(f^s(s)) = \{a_1 \in Act_1 | a_1 \in \omega(f^s(s))\}$.

By the correctness of a permissive policy for the safety specification φ, the following corollary holds. Intuitively, it is not possible for Agent 1 to pick an "unsafe" action.

Corollary 1 (Safety)**.** *In the restricted POSG $\mathcal{G}^{\omega}$, all policies for Agent 1 satisfy the safety specification φ.*

It remains to project $\mathcal{G}^{\omega}$ to the POMDP using the reward-relevant features as in Def. 1. According to Theorem 1, a reward-maximizing policy (if it exists and can be computed) will satisfy the safety specification and solve our problem.

IV. Case Studies

A. Restricted UAV Delivery

We refer to the introduction (Section I-A) for a high-level description of this case study. Partial observability arises from the stochastic initial launch location of the UAV. In a gridworld of size m, Agent 1 is uncertain of its position (x_a, y_a).

Observations: We refer back to Fig. 1, and consider Agent 1 at a state $s \in S_1$ with viewing range n. The observation model $O(s)$ returns distinct observations A_1 and A_2 when flying over the target areas. Recall that we assume two feature spaces X and Y, where X describes the physical location of the UAV, and Y describes its relative position to the dispatcher. We look at state $s = (x, y, 1)$ with $x = x^r = (x_a, y_a)$ and dispatcher position (x_d, y_d). Then, we have $y = (x_a - x_d, y_a - y_d)$.

We define the set W of states that ensure the UAV is *within the viewing window* of the dispatcher as $W := \{(x, y, 1) \in S_1 \mid |x_a - x_d| \leq n \wedge |y_a - y_d| \leq n)\}$. Similarly, the set $C := \{(x, y, 1) \in S_1 \mid (x_a = x_d) \wedge (y_a = y_d)\}$ describes a *collision* of the UAV and the dispatcher.

Specifications: In this setting, the safety specification is given by $\varphi_s = \neg\Diamond\,(\neg W \wedge C)$, that is, Agent 1 and Agent 2 are never out of visible range and never collide. Furthermore, Agent 1 seeks to minimize the expected number of steps required to first visit the green headquarters at A_1 and then to the purple construction site at A_2, $\psi_r = \Diamond\,(A_1 \wedge \Diamond\, A_2)$.

Decomposition: Our *reward feature projection* $f^r(s) = x^r = x = (x_a, y_a)$ maps the state $s \in S_1$ to feature x that corresponds to the UAV location, as only the UAV location is relevant. The *safety-feature projection* $f^s(s) = (y, \cdot)$ maps the state s to $(y, \cdot)$ where $y = (x_a - x_d, y_a - y_d)$ without the need for explicit knowledge on the value of x_a, x_d, y_a and y_d. After separation, the POMDP problem closely resembles that of the 4x4 grid in [27] with multiple goal locations.

B. Mars Rover with Adversarial Charging

Inspired by *Ingenuity*, the paired helicopter with NASA's *Perseverance* rover [36], we consider a modified version of the well-known *RockSample* problem [40], which itself was inspired by the *Spirit/Opportunity* mission. While *Ingenuity* relies on solar power, we consider a potential scenario where it needs to periodically draw charge from the larger rover that is performing other higher priority missions. We model *Perseverance* (Agent 2) as an adversary that moves the charging location, and *Ingenuity* (Agent 1) must plan accordingly.

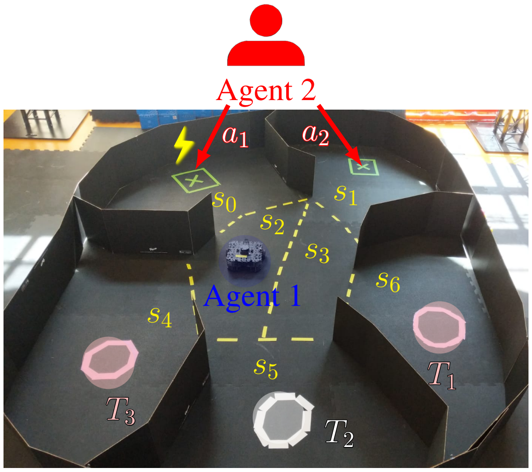


Fig. 4: Environment for case study with rock sampling robot (Agent 1 - blue) and charging allocator (Agent 2 - red) that decides active the charging station (green). Agent 2 may select the active charging location ($Act_2 = \{a_1, a_2\}$) every 5 time steps. Agent 1 receives reward $R = +25$ for sampling the pink targets $\{T_1, T_3\}$ and receives reward $R = -25$ for sampling the white target T_1. Each action choice incurs an $R = -1$ energy penalty. While in the same room, Agent 1 may make a noisy observation on the status of a target. Agent 1 attempts to accumulate the maximum expected reward while ensuring that it will never be out of charge due to the actions of Agent 2.

Our modified problem setting involves a robotic object-sampling task with Agent 2 controlling the active charging location (see Fig. 4). Refer also to Fig. 6 where we created a series of video still that show the behavior of the permissive policy for the resulting projected POMDP.

In *RockSample*, Agent 1 attempts to determine which rocks are valuable and take samples of valuable rocks. In this experiment, we physically denote the *Good* rocks (those producing positive reward $R = +25$) as pink targets and the *Bad* rocks (with reward $R = -25$) as white targets. Fig 4 has two *Good* targets $\{T_1, T_3\}$ from a set of three candidate objects $\{T_1, T_2, T_3\}$ across three rooms $\{s_4, s_5, s_6\}$. For discretized planning, we abstract the environment into seven "rooms" - two of which contain the charging locations, three contain targets with potential reward and two are waypoints that one may pass through to reach these locations. Time intervals are segmented into the time taken to travel from one room to another, and power draw is equal across time intervals (each action accumulates $R = -1$). Additionally, we replace the map reward in *RockSample* with a power constraint, i.e. Agent 1 has a finite power supply and must recharge every 5 time-steps. Periodically, Agent 2 decides which one of the two charging points (green in Fig 4) is active $Act_2 = \{a_1, a_2\}$. Agent 1's actions Act_1 are defined by taking choosing one of the following:

- Move to an adjacent room,

- Remain in existing room,
- Check status of object in current room (noisy sensor),
- Sample object in current room to collect potential reward,
- Shut down to end experiment.

The state space $S = L \times E \times A \times \mathcal{T}$ is the product of the following 4 features: robot location $L = \{s_0, \cdots, s_6\}$, charge level $E = \{0, \cdots, 5\}$, active charging location $A = \{A_1, A_2\}$ and target type $\mathcal{T} = \{T_1, T_2, T_3\}$.

Observations: In this setting, observations are received by performing the action $Check_i \in Act_1(s)$ for activating the robot's sensor. This action is only available to Agent 1 when it occupies the same room as the target it is checking. The observation function is thus a mapping $O\colon \mathcal{T} \to Distr(Z)$ where $Z = \{Bad, Good\}$. Unlike the original *RockSample*, the noisy sensor has a fixed detection probability $p = 0.8$.

Specifications: Agent 2 may switch charging locations every 5 time-steps. Therefore Agent 1 must ensure that it can safely reach a charging location before it is switched, which can be described as $\varphi_E = \neg\Diamond\,(e = 0)$.

Decomposition: The *reward feature projection* $f^r(s) = x^r = (s_i, t_1, t_2, t_3)$ maps the state $s \in S_1$ to feature x^r, a tuple of robot location and target statuses. The *safety feature projection* $f^s(s) = (x^s, y) = ((s_i, e), A_j)$ maps the state $s \in S_1$ to feature x^s, a tuple of robot location and energy status, and feature y, the active charging location $A_j \in A$. After separation, the POMDP synthesis problem closely resembles an instance of *RockSample* with slightly different reward conditions.

C. Autonomous Car with Sensor Array

Consider the scenario, pictured in Fig. 5, of an autonomous vehicle operating in a city with the following sensors:

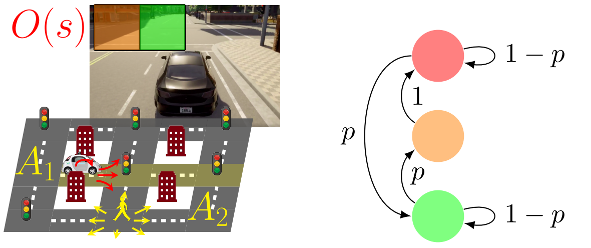
1) Navigation System (GNSS),
2) Optical Camera (Traffic Light Identification),
3) Lidar (Proximity),
4) Radar (Speed).

At each intersection is a set of traffic lights that restrict the available action choices of the vehicle:

- Red: $\{Stop\}$,
- Yellow: $\{Stop, Straight\}$,
- Green: $\{Stop, Straight, Left, Right\}$.

The state of each traffic light can be modeled as a Markov chain where the system switches from red to green and green to orange with probability p, see Fig. 5b. Agent 2 in this game is a pedestrian, who moves with greater flexibility but reduced speed. Each turn, the vehicle takes two actions while the pedestrian can move one intersection at a time. While the vehicle must move according to the status of the lights, the pedestrian is under no such restriction and will ignore them.

State space: A state in this system is $s = (x, y, \theta, obs_x, obs_y, L)$, which is the location and direction of the vehicle and the location of the pedestrian as well as the status of the lights L at each intersection.



(a) Environment (below) and a sample observation (above).

(b) Traffic light transitions with probability p.

Fig. 5: Autonomous car (Agent 1) operating in an adversarial urban environment modeled as a gridworld and traffic lights. Each intersection has a decision point for Agent 1 who moves twice for each pedestrian move (Agent 2).

Partial observability: The car is only able to see the traffic lights that are directly in front of it. For example in Fig. 5a, the vehicle facing to the right can observe the lights of two intersections (the middle and the center right). The vehicle uses its optical sensors to determine the status of the lights. Without the pedestrian, the task of navigating from an initial position A_1 to a goal location A_2 is modeled as a POMDP, with a belief on each status of the unseen intersection lights.

Safety features: In this case study, we demonstrate the effect of different specifications on the feature splitting. In particular, we describe two scenarios where the stochastic game problem differs based on the safety task and the sensors that autonomous vehicle uses to measure the relevant features.

- **Pedestrian never follows car** - in this mission the pedestrian attempts to read the number plate of the back of the car. To perform this mission, the pedestrian must be behind the car T for two consecutive turns - described by $\varphi_F = \neg\Diamond\,(T \wedge \bigcirc T)$. Note that the $\bigcirc$-operator in temporal logic refers to the "next" state. The vehicle uses the lidar proximity sensor to construct the relevant feature that determines its relative position to the pedestrian.
- **Vehicle cannot speed within vicinity of pedestrian** - since the vehicle moves twice for every one action cycle, we define "speeding" as moving straight twice through the green lights. For this specification, we utilize a lidar to sense the relative position as well as the radar to test speed. $\varphi_G = \neg\Diamond\,((a_1 \wedge \bigcirc a_1) \wedge M)$ where M is when the pedestrian within one space of the vehicle. The direction of the vehicle is not relevant to the specification since M is only a function of the adjacency of the pedestrian.

D. Implementation and Results

Experimental Setup: Although the proposed approach is amenable to any POMDP solver, we use the POMDP module of the tool Storm [8]. PRISM-POMDP [31] and SolvePOMDP [44] are other tools that could perform this function. For computing permissive strategies we implement the problem as an MILP [10, 21] and solve using Gurobi

TABLE I: Synthesis times for POSG by subcomponent

Setting				POMDP		SG		POSG
Case	(m,n)	Spec.	Size (states)	Time (s)	Result $V^{\sigma_1}(b_0)$	Size (states)	Time (s)	Time (s)
UAV Delivery	(5,3)	φ_W	1729	150.93	12.35	8	0.08	151.01
UAV Delivery	(5,4)	φ_W	—"—	—"—	—"—	15	0.13	151.06
UAV Delivery	(5,3)	φ_W and σ_2	5450	402.93	26.11	-NA-	-NA-	-NA-
UAV Delivery	(15,5)	φ_W	37065	823.12	42.04	24	0.23	823.36
UAV Delivery	(15,5)	φ_W and σ_2	503930	6942.63	67.12	-NA-	-NA-	-NA-
UAV Delivery	(25,7)	φ_W	165529	3077.10	62.90	48	0.81	3077.91
UAV Delivery	(25,7)	φ_W and σ_2	-TO-	-TO-	-TO-	-NA-	-NA-	-NA-
UAV Delivery	(40,7)	φ_W	663229	25301.00	103.83	—"—	—"—	25301.81
Assured Charging Robot	-NA-	φ_E	56	1.32	14.96	84	0.12	1.44
Autonomous Vehicle	(3)	φ_F	26244	350.23	6.42	1048	5.31	355.54
Autonomous Vehicle	(3)	φ_G	—"—	—"—	—"—	226	1.25	351.48
Autonomous Vehicle	(3)	φ_F and σ_2	944784	-TO-	-TO-	-NA-	-NA-	-NA-
Autonomous Vehicle	(4)	φ_G	262144	5032.91	10.38	—"—	—"—	5034.16

9.0.1 [12]. All policy computations were performed on a 8-core 1.9 GHz machine with a 12 GB memory limit and a time limit of 10^5 seconds.

For the physical experiment, we used a TurtleBot 3 Waffle Pi robot connected to the Robot Operating System (ROS) to implement the pre-computed (randomized) policy. Target observations were simulated using a prescribed probability distribution outlined in Section IV-B.

For the high-fidelity autonomous vehicle simulation, we implement the computed policies on the open-source simulator CARLA [9]. We run CARLA 0.9.10 on a 3.1 GHz machine with a GeForce RTX2060 graphics and 32GB of memory.

Policy Computation: Table I contains a breakdown of the model sizes, running times, and expected values of the various components of the decomposition process. Each case study was designed to be scalable for parameters m and n, where respective parameters were: grid size and window size in UAV delivery; and grid size in autonomous vehicle. We performed the following experiments.

- **SG** is the running time to compute a permissive policy on the projected stochastic game as in Definition 2.
- **POMDP** is the running time to solve the projected POMDP as in Definition 1, that is, to compute the reward-optimal policy restricted to the permissive policy resulting from the safety-feature projection.
- **POSG** is an experiment where we fix a policy for Agent 2, that is, all actions are chosen according to a uniform distribution. The resulting model is effectively a POMDP and amenable to standard solvers. The resulting POMDP is not restricted with respected to the permissive policy, and may be significantly larger than the POMDP that is guaranteed to adhere to safety specifications.

The cumulative time of "POMDP" and "SG" is the overall running time of our method. Note that we are able to solve **POSGs with hundreds of thousands of states** within the time limit. These model sizes are too large for a feasible comparison with existing POSG methods, however, we demonstrate the scalability of our approach using the POSG experiment However, without the restriction via the permissive policy, these POMDPs are large and take longer to solve than our two-stage approach.

Discussion: In the multimedia appendix, one can see how the permissive policy modifies the available choices for Agent 1. We highlight a frame-by-frame walkthrough of a video regarding the Mars rover with adversarial charging in the appendix within Fig. 6. Specifically, in Fig 4, when the active charging location is at A_1 and Agent 1 has moved from location s_4 to location s_2 (energy level low at $e = 1$), the permissive policy only allows for one transition - back to s_0 with the active A_1. Similarly in the first instance (active charge A_1), Agent 1 needs three actions to reach T_1 at s_6 and then three to return to charge. Therefore the permissive policy rules out the transition from s_3 to s_6. The POMDP solver takes that into account and therefore sees no utility gained by going from s_2 to s_3 (refer again to Fig. 6).

In the autonomous driving simulator, the permissive policy often just constrains the Agent 1's ability to go straight through an intersection forcing Agent 1 into more uncertain routes. See Fig. 7 for a visual description of how the permissive policy changes the nominal path of the autonomous vehicle.

V. Conclusion

We presented a new scheme to solve POSGs in a multi-objective setting that captures typical trade-offs between safety and performance of an agent and its environment. We demonstrated that there are realistic examples that naturally fit into the setting and the necessarily strong assumptions we have to make. In the future, we will investigate automatic ways, for instance, based on structure learning and influence diagrams [35], to identify and separate the relevant features.

Acknowledgments

This research has been partially supported by ONR N00014-19-1-2054, ARL ACC-APG-RTP W911NF, ONR N00014-18-1-2829 and NWO OCENW.KLEIN.187.

References

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe Reinforcement Learning via Shielding. In *AAAI Conf. on Artificial Intelligence*, pages 2669–2678. AAAI Press, 2018.

[2] Paul William Berg, Scott Isaacs, and Kelsey Lynn Blodgett. Airborne Fulfillment Center Utilizing Unmanned Aerial Vehicles for Item Delivery, April 5 2016. US Patent 9,305,280.

[3] Suda Bharadwaj, Roderik Bloem, Rayna Dimitrova, Bettina Könighofer, and Ufuk Topcu. Synthesis of Minimum-Cost Shields for Multi-agent Systems. In *ACC*, pages 1048–1055. IEEE, 2019.

[4] Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.*, 78(2):394–413, 2012.

[5] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. A Survey of Partial-observation Stochastic Parity Games. *Formal Methods Syst. Des.*, 43(2):268–284, 2013.

[6] Krishnendu Chatterjee, Laurent Doyen, Hugo Gimbert, and Thomas A. Henzinger. Randomness for Free. *Inf. Comput.*, 245:3–16, 2015.

[7] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On Stochastic Games with Multiple Objectives. In *MFCS*, volume 8087 of *LNCS*, pages 266–277. Springer, 2013.

[8] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *CAV (2)*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.

[9] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *CoRL*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 2017.

[10] Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Permissive Controller Synthesis for Probabilistic Systems. In *TACAS*, volume 8413 of *LNCS*, pages 531–546. Springer, 2014.

[11] Rosemary Emery-Montemerlo, Geoffrey J. Gordon, Jeff G. Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, pages 136–143. IEEE Computer Society, 2004.

[12] LLC Gurobi Optimization. Gurobi Optimizer Reference Manual, 2020.

[13] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI Conf. on Artificial Intelligence*, pages 709–715. AAAI Press, 2004.

[14] Milos Hauskrecht. Value-function Approximations for Partially Observable Markov Decision Processes. *J. Artif. Intell. Res.*, 13:33–94, 2000.

[15] Karel Horák and Branislav Bosanský. Solving Partially Observable Stochastic Games with Public Observations. In *AAAI Conf. on Artificial Intelligence*, pages 2029–2036. AAAI Press, 2019.

[16] Karel Horák, Branislav Bosanský, and Michal Pechoucek. Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games. In *AAAI Conf. on Artificial Intelligence*, pages 558–564. AAAI Press, 2017.

[17] Karel Horák, Quanyan Zhu, and Branislav Bosanský. Manipulating Adversary's Belief: A Dynamic Game Approach to Deception by Design for Proactive Network Security. In *GameSec*, volume 10575 of *LNCS*, pages 273–294. Springer, 2017.

[18] Karel Horák, Branislav Bosanský, Christopher Kiekintveld, and Charles A. Kamhoua. Compact Representation of Value Function in Partially Observable Stochastic Games. In *IJCAI*, pages 350–356. AAAI Press, 2019.

[19] Shin Ishii, Hajime Fujita, Masaoki Mitsutake, Tatsuya Yamazaki, Jun Matsuda, and Yoichiro Matsuno. A Reinforcement Learning Scheme for a Partially-Observable Multi-Agent Game. *Mach. Learn.*, 59(1-2):31–54, 2005.

[20] Nils Jansen, Bettina Könighofer, Sebastian Junges, and Roderick Bloem. Shielded Decision-Making in MDPs. *CoRR*, abs/1807.06096, 2018.

[21] Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-Constrained Reinforcement Learning for MDPs. In *TACAS*, volume 9636 of *LNCS*, pages 130–146. Springer, 2016.

[22] Sammie Katt, Frans A. Oliehoek, and Christopher Amato. Bayesian Reinforcement Learning in Factored POMDPs. In *AAMAS*, pages 7–15. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[23] Bettina Könighofer, Mohammed Alshiekh, Roderick Bloem, Laura R. Humphrey, Robert Könighofer, Ufuk Topcu, and Chao Wang. Shield synthesis. *Formal Methods in System Design*, 51(2):332–361, 2017.

[24] Vojtech Kovarík, Martin Schmid, Neil Burch, Michael Bowling, and Viliam Lisý. Rethinking Formal Models of Partially Observable Multiagent Decision Making. *CoRR*, abs/1906.11110, 2019.

[25] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Trans. Robotics*, 25(6):1370–1381, 2009.

[26] Akshat Kumar and Shlomo Zilberstein. Dynamic Programming Approximations for Partially Observable Stochastic Games. In *FLAIRS Conference*. AAAI Press, 2009.

[27] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. In *ICML*, pages 362–370. Morgan Kaufmann, 1995.

[28] Omid Madani, Steve Hanks, and Anne Condon. On the Undecidability of Probabilistic Planning and Infinite-

Horizon Partially Observable Markov Decision Problems. In *AAAI Conf. on Artificial Intelligence*, pages 541–548. AAAI Press, 1999.

[29] Weichao Mao, Kaiqing Zhang, Erik Miehling, and Tamer Basar. Information State Embedding in Partially Observable Cooperative Multi-Agent Reinforcement Learning. In *CDC*, pages 6124–6131. IEEE, 2020.

[30] João V. Messias, Matthijs T. J. Spaan, and Pedro U. Lima. Efficient Offline Communication Policies for Factored Multiagent POMDPs. In *NIPS*, pages 1917–1925, 2011.

[31] Gethin Norman, David Parker, and Xueyi Zou. Verification and Control of Partially Observable Probabilistic Systems. *Real-Time Systems*, 53(3):354–402, 2017.

[32] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. POMDPs for robotic tasks with mixed observability. In *Robotics: Science and Systems*. The MIT Press, 2009.

[33] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, pages 1025–1032. Morgan Kaufmann, 2003.

[34] Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977. doi: 10.1109/SFCS.1977.32.

[35] Kyle Polich and Piotr J. Gmytrasiewicz. Interactive dynamic influence diagrams. In *AAMAS*, page 34. IFAAMAS, 2007.

[36] Ned Potter. A Mars helicopter preps for launch: The first drone to fly on another planet will hitch a ride on NASA's Perseverance rover. *IEEE Spectrum*, 57(7):06–07, 2020.

[37] John H. Reif. Universal games of incomplete information. In *STOC*, pages 288–308. ACM, 1979.

[38] Stuart J. Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach (3. ed.)*. Pearson Education, 2010.

[39] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *NIPS*, pages 2164–2172, 2010.

[40] Trey Smith and Reid G. Simmons. Heuristic Search Value Iteration for POMDPs. *CoRR*, abs/1207.4166, 2012.

[41] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *J. Artif. Intell. Res.*, 24:195–220, 2005.

[42] Alexander L Strehl, Carlos Diuk, and Michael L Littman. Efficient structure learning in factored-state MDPs. In *AAAI Conf. on Artificial Intelligence*, volume 7, pages 645–650, 2007.

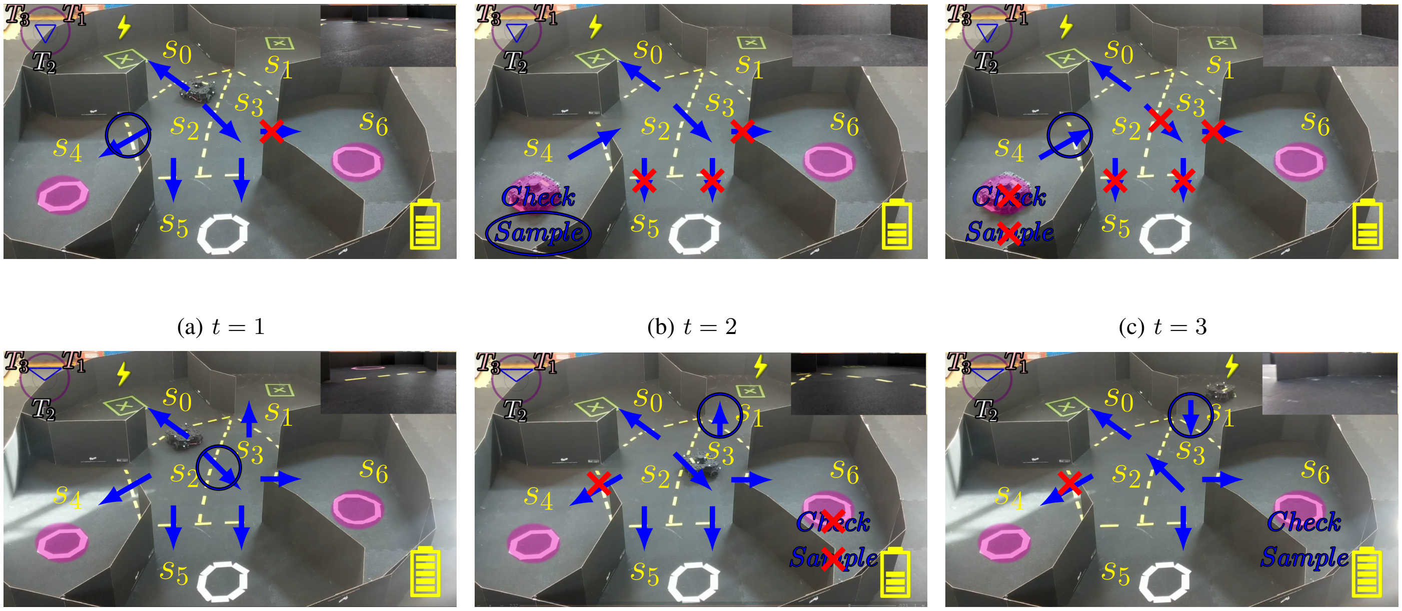
[43] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *JMLR*, 15(1):3483–3512, 2014.

[44] Erwin Walraven and Matthijs T. J. Spaan. Accelerated Vector Pruning for Optimal POMDP Solvers. In *AAAI Conf. on Artificial Intelligence*, pages 3672–3678. AAAI Press, 2017.

[45] Jason D Williams, Pascal Poupart, and Steve Young. Factored Partially Observable Markov Decision Processes for Dialogue Management. In *Proc. IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 76–82, 2005.

Appendix

In the appendix we provide labeled screenshots of the Mars rover adversarial charging (physical) experiment and the high-fidelity simulation of the autonomous driving task.



(a) $t = 1$

(b) $t = 2$

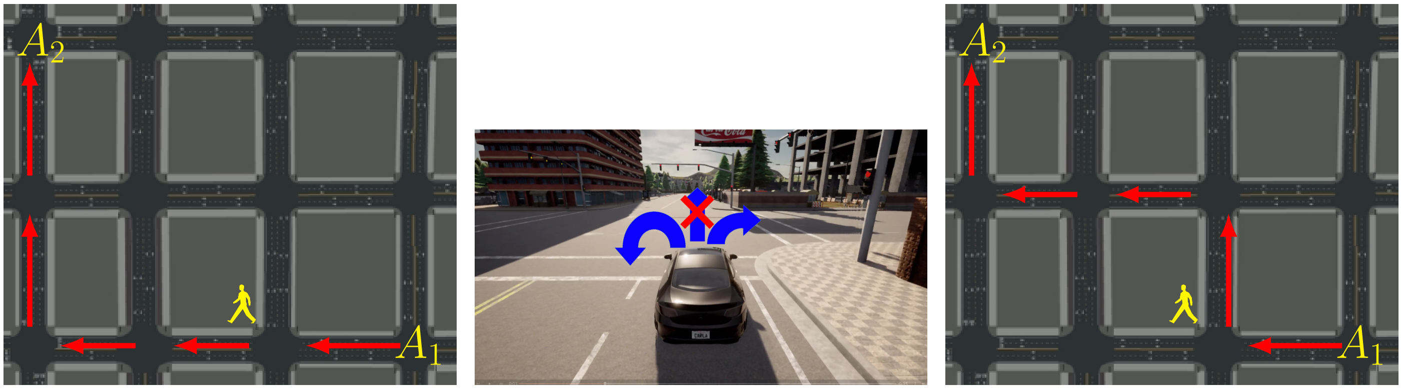
(c) $t = 3$

(d) $t = 26$

(e) $t = 27$

(f) $t = 28$

Fig. 6: A sequence of video stills for the Mars Rover case study, showing allowed actions via the permissive policy at different time steps.



(a) Nominal path in Carla environment

(b) Permissive policy at $t = 1$

(c) Modified path in Carla environment

Fig. 7: Change in the nominal path for the autonomous vehicle case study. The safety specification φ_G restricts Agent 1 from speeding past the pedestrian (Agent 2). Accordingly, Agent 1 takes a modified path through the Carla environment.