Safe POMDP Online Planning via Shielding

Shili Sheng[1], David Parker[2], and Lu Feng[1]

***Abstract*— Partially observable Markov decision processes (POMDPs) have been widely used in many robotic applications for sequential decision-making under uncertainty. POMDP online planning algorithms such as Partially Observable Monte-Carlo Planning (POMCP) can solve very large POMDPs with the goal of maximizing the expected return. But the resulting policies cannot provide safety guarantees which are imperative for real-world safety-critical tasks (e.g., autonomous driving). In this work, we consider safety requirements represented as almost-sure reach-avoid specifications (i.e., the probability to reach a set of goal states is one and the probability to reach a set of unsafe states is zero). We compute shields that restrict unsafe actions which would violate the almost-sure reach-avoid specifications. We then integrate these shields into the POMCP algorithm for safe POMDP online planning. We propose four distinct shielding methods, differing in how the shields are computed and integrated, including factored variants designed to improve scalability. Experimental results on a set of benchmark domains demonstrate that the proposed shielding methods successfully guarantee safety (unlike the baseline POMCP without shielding) on large POMDPs, with negligible impact on the runtime for online planning.**

I. Introduction

Partially observable Markov decision processes (POMDPs) provide a general framework for sequential decision-making under uncertainty and have been widely used in many robotic applications [1], [2]. For example, POMDP modeling and planning have been applied to robot localization and navigation [3], autonomous driving [4], human-robot interaction [5], and multi-robot coordination [6]. POMDP online planning is a paradigm where the policy computation and execution are interleaved: the agent computes an optimal action based on the current belief state, executes the action, receives an observation, and continues by computing an action for the resulting new belief state. Online planning can scale up to solve larger POMDPs than offline planning where a policy is computed for all possible belief states before execution [7].

Modern sampling-based algorithms (e.g., POMCP [8], DESPOT [9]) further improve the scalability of POMDP online planning by representing belief states as collections of particles without explicit belief state tracking by Bayes filtering. The goal of these algorithms is to compute (approximately) optimal policies that maximize the expected return. But the resulting policies cannot provide safety guarantees (e.g., never visit unsafe states), which are imperative for real-world safety-critical robotic tasks (e.g., autonomous driving).

Prior work has proposed various ways to account for (safety) constraints within POMDP online planning. The cost-constrained POMCP algorithm [10] seeks to compute optimal actions that maximize the reward while constraining the cost. A constant-horizon planning method is developed in [11] for chance-constrained POMDPs (i.e., maximizing the cumulative expected reward such that the probability of ending up in risky states is at most δ). An online method is proposed in [12] for synthesizing partial conditional plans that satisfy safe-reachability objectives (i.e., reaching a goal state with probability greater than δ_1 while keeping the probability of visiting unsafe states below δ_2).

In this work, we consider stricter safety requirements represented as *almost-sure reach-avoid specifications* [13], where the probability to reach a set of goal states is one and the probability to reach a set of unsafe (avoid) states is zero. We construct shields that restrict unsafe actions based on pre-computed winning regions (i.e., sets of POMDP belief support states) that satisfy almost-sure reach-avoid specifications. Then, we integrate these shields with the POMCP algorithm for safe POMDP online planning.

We propose four distinct shielding methods, differing in how the shields are constructed and integrated. In *centralized shielding*, we construct a shield based on the maximal winning region of the entire POMDP model. By contrast, in *factored shielding*, we decompose a large POMDP into a set of smaller models based on the factorization of the state space and compute winning regions for each factored model separately. We shield actions for the POMCP algorithm via either *prior pruning* (checking if any action of the current belief state should be shielded) or *on-the-fly backtracking* (checking every action encountered during simulation).

We evaluate the proposed methods via experiments on a set of benchmark domains. The experimental results show that the proposed shielding methods can guarantee safety, while policies computed by the baseline (POMCP without shielding) cannot avoid unsafe states completely. Moreover, the search time per planning step of our shielding methods is comparable with the baseline. Factored shielding methods demonstrate better scalability than centralized shielding methods. We observe that on-the-fly backtracking generally yields higher expected return than prior pruning.

II. Related Work

Our work is inspired by [13], where winning regions are computed to enforce almost-sure reach-avoid specifications in POMDPs. In recent work [14], these are then used

[1]Shili Sheng and Lu Feng are with the School of Engineering and Applied Science, University of Virginia, Charlottesville, VA 22904, USA {ss7dr, lf9u}@virginia.edu

[2]David Parker is with the Department of Computer Science, University of Oxford, Parks Road, Oxford OX1 3QD, United Kingdom david.parker@cs.ox.ac.uk

for shielding in safe reinforcement learning. We apply the methods of [13] to compute winning regions and consider shielding in the context of POMDP online planning. We also develop a new algorithm of computing factored winning regions for solving large POMDPs.

A rule-based shielding of the POMCP algorithm is presented in [15], where shields are obtained by learning parameters for a set of rule templates defined by experts. An example rule template is "the robot should move fast if it is highly confident that the aisle in which it is moving is not cluttered". By analyzing belief-action traces previously generated by the agent, the rule template is instantiated as "the robot usually moves fast if the probability of being in a cluttered segment is lower than 7%". The learnt rules are then used as a shield to preemptively prune undesired actions considering the current belief state, similar to our prior pruning procedure.

The safe-reachability objectives considered in [12] are quantitative variants of reach-avoid specifications where the probability to reach goals or avoid unsafe states should be bounded within certain thresholds. An online synthesis method is presented in [12] to compute a *partial conditional plan* (which only contains a sampled subset of all possible events and approximates a full POMDP policy) subject to safe-reachability objectives. But this method does not account for the expected return.

There is also a line of related work on online planning for constrained POMDPs. Online algorithms such as [16], [10] consider cost constraints to bound the expected cumulative costs, but they cannot completely prevent actions that violate the cost constraint. The chance-constrained POMDP problem that seeks to bound the probability of failure is reduced to a cost-constrained problem in [11]. The online planning method *expectation optimization with probabilistic guarantee* (EOPG) is considered in [17], where the objective is to maximize the expected return with respect to all policies that ensure at least τ payoff with probability at least δ. Earlier work [18] studies a similar problem named *guaranteed payoff optimization* (i.e., EOPG with $\delta = 1$). These prior approaches have different objectives from our problem and thus are not directly comparable with our work.

III. Background

We denote by $\mathbb{R}$ the set of reals, and $Dist(X)$ the set of probability distributions over a random variable X.

A. POMDP Model

We denote a POMDP model as a tuple (S, A, O, T, R, Z), where S, A and O are (finite) sets of states, actions, and observations, respectively; $T : S \times A \to Dist(S)$ is the transition function where $T(s, a, s') = Pr(s'|s, a)$ denotes the probability of ending in state s' when taking action a in state s; $R : S \times A \to \mathbb{R}$ is the reward function; and $Z : S \times A \times O \to [0, 1]$ is the observation function where $Z(s', a, o) = Pr(o|s', a)$ represents the probability of observing o after taking action a and ending in state s'.

Since POMDP states are partially observable, the agent keeps track of a *history* of actions and observations, denoted by $h_t = \{a_0, \ldots, a_{t-1}, o_t\}$, and chooses an action a_t at time t following a *policy* π that maps h_t to $Dist(A)$. A policy is *deterministic* if it always picks a Dirac distribution. For POMDP online planning that seeks to maximize the expected return, it suffices to only consider deterministic policies [7].

A *belief* state at time t represents the posterior probability distribution of being in each state given the history, denoted by $b_t(s) = Pr(s_t = s|h_t)$ for $s \in S$. The initial belief state b_0 represents a distribution over initial states of the POMDP. The belief b_t at time t can be obtained via a belief update function $b_t = \tau(b_{t-1}, a_{t-1}, o_t)$ following Bayes' rule. We denote by $Supp(b) := \{s \in S|b(s) > 0\}$ the *belief support* of a belief b. Let $Pr_b^\pi(T)$ denote the probability to reach a set $T \subseteq S$ of states from belief b under policy π.

B. Almost-Sure Reach-Avoid Specifications

We consider *almost-sure reach-avoid* specifications [13], denoted by $\varphi = \langle \mathsf{REACH}, \mathsf{AVOID} \rangle \subseteq S \times S$ with $\mathsf{REACH} \cap \mathsf{AVOID} = \emptyset$. We say that a POMDP policy π is *winning* for φ from belief b iff $Pr_b^\pi(\mathsf{AVOID}) = 0$ and $Pr_b^\pi(\mathsf{REACH}) = 1$, that is, starting from belief b and under policy π, the AVOID set is reached with probability zero while the REACH set is reached with probability one (almost surely).

We define a *winning region* W_φ for an almost-sure reach-avoid specification φ as a set of belief supports where every belief support $Supp(b) \in W_\varphi$ is *winning* (i.e., there exists a winning policy π for φ from b). The *maximal winning region* for φ is the region containing all winning belief supports. A winning region W_φ is called *productive* if, from every belief support $Supp(b) \in W_\varphi$, there exists a (finite) path to stay within the region and reach some state in REACH set. An agent that stays within a productive winning region W_φ is guaranteed to satisfy the almost-sure reach-avoid specification φ. We can apply the SAT-based iterative approach in [13] to compute productive subsets of the maximal winning region.

C. Partially Observable Monte-Carlo Planning

In this work, we adopt a widely used POMDP online planning algorithm named *Partially Observable Monte Carlo Planning* (POMCP) [8]. At each time step t, the POMCP algorithm uses Monte Carlo tree search [19] to explore a search tree whose root node is denoted by $\mathcal{T}(h_t) = \langle N(h_t), V(h_t), \beta(h_t) \rangle$, where $N(h_t)$ counts the number of times that history h_t has been visited, $V(h_t)$ estimates the expected return of all simulations starting with h_t, and $\beta(h_t)$ is a set of particles (each of which corresponds to a POMDP state) as an approximation of belief b_t. The algorithm repeats the following four phases.

1) **Selection:** Randomly sample a state s from $\beta(h_t)$.
2) **Expansion:** Once a leaf node $\mathcal{T}(h)$ is reached, expand the search tree with child nodes $\mathcal{T}(ha) = \langle N_{init}(ha), V_{init}(ha), \emptyset \rangle$ for all actions $a \in A$.
3) **Simulation:** For each history h encountered during the simulation, choose an action a that maximizes

$V(ha) + c\sqrt{\frac{logN(h)}{N(ha)}}$ following the *upper confidence bound* (UCB) rule for balancing between exploration and exploitation if $\mathcal{T}(h)$ is a non-leaf node; otherwise, choose an action a following a rollout policy (e.g., uniform random action selection). Then, use a black box simulator $(s', o, r) \sim \mathcal{G}(s, a)$ to generate a successor state s' and add it to the particle set $\beta(hao)$. The simulation continues with s' as the start state until the simulated path attains a target depth.

4) **Backpropagation:** Use the information obtained from the simulation to update the nodes (i.e., N counts, V values, β particles) along the path from the root node to the leaf node in the search tree.

The planning for time step t ends when a target number of iterations of the above four phases have been completed (or a timeout elapses). Then, the agent executes the optimal action $a_t = \arg\max_a V(h_t a)$ and receives an observation o_{t+1}. The algorithm continues the online planning for the next time step with a search tree rooted from node $\mathcal{T}(h_t a_t o_{t+1})$.

The POMCP algorithm scales well because it breaks the *curse of dimensionality* (by sampling states from a particle set) and *the curse of history* (by sampling histories using a black box simulator).

IV. Motivating Example

Consider an example of a robot navigating in the grid world environment shown in Figure 1. Let g_{ij} denote the grid location in row i and column j. We model the environment as a POMDP with the state space $S = \{g_{ij}\}$ for $1 \leq i, j \leq 6$. The robot can take four actions to move *east*, *south*, *west*, and *north*, respectively. For each action, the robot moves to the target location with probability 0.8 or overshoots by one location with probability 0.2 due to slippery roads. The robot starts in g_{11} and aims to reach the flag in g_{66} for which it will receive a reward of 1,000. The step cost is 1 and the cost of colliding with an obstacle is 5. The robot's location during the trip is uncertain due to noisy sensors. Thus, we use POMDP belief states to represent the posterior probability distribution of the robot being in each grid given the history, with the initial belief $b_0(g_{11}) = 1$.

Applying the POMCP algorithm to the above example yields one possible trajectory of the robot: $g_{11} \xrightarrow{east} g_{13} \xrightarrow{east} g_{15} \xrightarrow{east} g_{16} \xrightarrow{south} g_{26} \xrightarrow{south} g_{46} \xrightarrow{south} g_{66}$. This trajectory exhibits unsafe behavior of the robot colliding into the obstacle in g_{15}.

In this work, we aim to tackle this problem by developing novel methods that can guarantee safety during the POMDP online planning.

V. Centralized Shielding

We consider safety requirements represented as almost-sure reach-avoid specifications (cf. Section III-B). Given a POMDP and an almost-sure reach-avoid specification φ, we compute a (large) productive winning region W_φ of the POMDP by applying the SAT-based iterative approach of [13] incrementally to a fixpoint. We define a *centralized shield* $\chi : b \to 2^A$ which, for any winning belief state b of the POMDP, gives *allowed* actions $\chi(b)$, which exclusively lead to belief support states within the winning region W_φ. In practice, we do not compute such a shield explicitly. Instead, we only store the winning region W_φ, and decide whether to allow any encountered action on-the-fly by checking if W_φ contains the resulting belief support states.

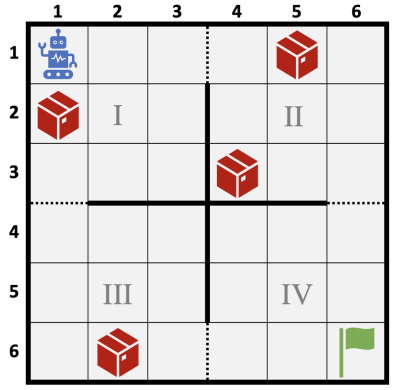

Fig. 1. An example grid world environment where the robot aims to reach the flag while avoiding obstacles. The robot can move through the doors (dashed lines) between four rooms that are separated by walls (solid lines).

We present two different ways of extending the POMCP algorithm with centralized shields for safe POMDP online planning, namely *prior pruning* and *on-the-fly backtracking*, in Sections V-A and V-B, respectively.

A. Prior Pruning

At each time step t, we find all actions disallowed by the shield $\chi(\beta(h_t))$ and prune the corresponding tree branches from the root node $\mathcal{T}(h_t)$ before the POMCP algorithm iterations. Specifically, for each action $a \in A$, we loop through every state $s \in \beta(h_t)$ and add successor states s' generated by a black box simulator $(s', o, r) \sim \mathcal{G}(s, a)$ to the set $\beta(h_t ao)$. We check if these belief support updates are contained in the winning region W_φ.

Following the motivating example, suppose that the history at time step 1 is $h_1 = \{east, g_{13}\}$, and the particle set is $\beta(h_1) = \{g_{12}, g_{13}\}$. Consider an almost-sure reach-avoid specification φ with $\mathsf{REACH} = \{g_{66}\}$ and $\mathsf{AVOID} = \{g_{15}, g_{21}, g_{34}, g_{62}\}$. We find that the shield $\chi(\beta(h_1))$ disallows action *east* because it may lead to a set of belief support states $\{g_{13}, g_{14}, g_{15}\}$ that are not contained in the winning region W_φ (since g_{15} is not a winning belief support state). We prune the tree branch of action *east* at node $\mathcal{T}(h_1)$. The unsafe robot trajectory in Section IV would be prevented.

Correctness. At each step t, any unsafe actions leading to non-winning belief support states are pruned. The agent selects the optimal action a_t that is expected to yield winning beliefs, in an approximate sense, in accordance with the almost-sure reach-avoid specification φ. It is important to note that, as POMCP is a sampling-based algorithm, there is a risk that approximate belief states might overlook unsafe states, leading to false positives. However, with a sufficiently large particle set, the POMDP policy derived from centralized shielding with prior pruning, can offer safety guarantees.

B. On-the-Fly Backtracking

The prior pruning approach can only shield actions at the root node $\mathcal{T}(h_t)$, without considering the safety of simulated paths, which may cause the value of $V(h_t a)$ to be estimated based on unsafe simulations. To address this limitation, we propose the following on-the-fly backtracking procedure.

During the simulation phase of the POMCP algorithm, when an action a is chosen (by the UCB rule or rollout) for history h and a successor state s' is generated by a black box simulator $(s', o, r) \sim \mathcal{G}(s, a)$, we check if the updated particle set $\beta(hao) \cup \{s'\}$ is contained in the winning region W_φ. If the resulting particle set is not winning, we prune the tree branch starting from node $\mathcal{T}(ha)$; that is, action a would be shielded at node $\mathcal{T}(h)$.

For example, suppose that state $s = g_{13}$ is sampled from the particle set $\beta(h_1) = \{g_{12}, g_{13}\}$ and action *east* is chosen. Suppose that the black box simulator yields a successor state $s' = g_{15} \in \mathsf{AVOID}$. Thus, we would shield action *east* at node $\mathcal{T}(h_1)$ and prevent the robot colliding into the obstacle.

Correctness. At each step t, the selected optimal action a_t has been encountered during the simulation and is allowed by a centralized shield via on-the-fly backtracking. Thus, the resulting POMDP policy is safe.

VI. Factored Shielding

The practical applicability of centralized shielding is limited by the computational effort required to obtain the winning region, which is correlated with the POMDP model size. To improve scalability, we develop a *factored* shielding method, where we decompose a POMDP model into a set of smaller submodels (see Section VI-A), compute a winning region for each submodel (see Section VI-B), and integrate the set of obtained winning regions into the POMCP algorithm (see Section VI-C). We show the correctness of the proposed method in Section VI-D.

A. Decomposing a POMDP Model

Given a POMDP model $\mathcal{M} = (S, A, O, T, R, Z)$, we decompose it into a set of N smaller POMDP models $\mathcal{M}^i = (S^i, A^i, O^i, T^i, R^i, Z^i)$ for $1 \leq i \leq N$ based on the factorization of the state space such that $S = \bigcup_i S^i$. We can leverage problem-specific knowledge to achieve an efficient decomposition scheme.

For example, we decompose the POMDP model $\mathcal{M}$ of the motivating example (see Section IV) into four submodels, corresponding to the four rooms shown in Figure 1. The state space S^1 of submodel $\mathcal{M}^1$ includes 9 grid locations covered by room I (i.e., $\{g_{ij}\}$ for $1 \leq i, j \leq 3$) and 4 outlet locations $\{g_{14}, g_{15}, g_{41}, g_{51}\}$ that can be reached by the robot when it takes an action in room I (e.g., moving east in g_{13} or moving south in g_{31}). We include these outlet locations as absorbing states in $\mathcal{M}^1$. The action space is $A^1 = A$. The set $O^1 \subseteq O$ only captures relevant observations of submodel states S^1. The transition function $T^1 : S^1 \times A^1 \to \mathit{Dist}(S^1)$, the reward function $R^1 : S^1 \times A^1 \to \mathbb{R}$, and the observation function $Z^1 : S^1 \times A^1 \to \mathit{Dist}(O^1)$ are projections of the original POMDP model's transition function T, reward function R, and observation function Z onto the submodel $\mathcal{M}^1$, respectively. We define submodels $\mathcal{M}^2$, $\mathcal{M}^3$ and $\mathcal{M}^4$ corresponding to rooms II, III and IV in a similar way.

B. Computing Factored Winning Regions

Given a set of factored POMDP models $\{\mathcal{M}^i\}_{i=1}^N$ and an almost-sure reach-avoid specification $\varphi = \langle \mathsf{REACH}, \mathsf{AVOID} \rangle$, we compute a set of winning regions $\{W_\varphi^i\}_{i=1}^N$ to encode factored shields following Algorithm 1.

First, for each model $\mathcal{M}^i$, we determine the set of initial states S^i_{init}, reach states S^i_{reach}, and avoid states S^i_{avoid}. Following the previous example, we define the initial states of $\mathcal{M}^1$ as $S^1_{init} = \{g_{11}, g_{12}, g_{13}, g_{31}\}$, including the robot's initial location g_{11} and three inlet locations that can be reached when the robot enters from an adjacent room. Note that we exclude the inlet location $g_{21} \in \mathsf{AVOID}$ from S^1_{init}. We initialize the set of reach states as $S^i_{reach} = \mathsf{REACH} \wedge S^i$ and the set of avoid states as $S^i_{avoid} = \mathsf{AVOID} \wedge S^i$. In this example, $\mathcal{M}^4$ is the only model initialized with a non-empty reach set $S^4_{reach} = \{g_{66}\}$.

We compute a productive winning region W^4_φ for the POMDP model $\mathcal{M}^4$ by applying the SAT-based iterative approach in [13] incrementally to a fixpoint. Since the robot may enter room IV directly from rooms II or III, we consider $\mathcal{M}^2$ and $\mathcal{M}^3$ as adjacent models of $\mathcal{M}^4$ and add pairs $\langle 2, 4\rangle, \langle 3, 4\rangle$ to a queue.

While the queue is non-empty, we remove the first element $\langle i, j\rangle$ of the queue and check if the reach set S^i_{reach} should be updated based on the winning region W^j_φ. Suppose that $\langle 2, 4\rangle$ is removed from the queue and $S^2_{reach} = \emptyset$. We find the winning region W^4_φ containing the outlet states $\{g_{46}, g_{56}\}$ of model $\mathcal{M}^2$ that the robot may encounter when moving from room II to room IV. We update the reach set as $S^2_{reach} = \{g_{46}, g_{56}\}$ and compute an updated winning region W^2_φ. Once a model's reach set and winning region are updated, we add all of its adjacent models to the queue. Algorithm 1 terminates when the queue is empty and returns the union of all factored winning regions $\bigcup_{i=1}^N W^i_\varphi$.

C. Shielding POMCP with Factored Winning Regions

We integrate the POMCP algorithm with factored winning regions via prior pruning or on-the-fly backtracking similar to the centralized shielding method described in Section V.

Factored shielding with prior pruning. At each step t, before the POMCP algorithm explore the tree with root node $\mathcal{T}(h_t)$, we compute $\beta(h_t ao)$ for each action $a \in A$ and prune any action that may lead to belief support states not contained in factored winning regions $\bigcup_{i=1}^N W^i_\varphi$.

Factored shielding with on-the-fly backtracking. During the POMCP simulation phase, we check if each successor state s' generated by a black box simulator yields winning belief support updates contained in $\bigcup_{i=1}^N W^i_\varphi$. We shield any action that leads to non-winning simulated beliefs.

Algorithm 1: Computing factored winning regions

Input: A set of factored POMDP models $\{\mathcal{M}^i\}_{i=1}^N$, an almost-sure reach-avoid specification φ

Output: A set of winning regions $\{W_\varphi^i\}_{i=1}^N$

determine $S_{init}^i, S_{reach}^i, S_{avoid}^i$ of each model $\mathcal{M}^i$
$queue \leftarrow [\,]$
foreach *model* $\mathcal{M}^j$ *with a non-empty set* S_{reach}^j **do**
    compute winning region W_φ^j
    foreach *adjacent model* $\mathcal{M}^k$ *of* $\mathcal{M}^j$ **do**
        $queue.put(\langle k, j\rangle)$
while *queue is non-empty* **do**
    $\langle i, j\rangle = queue.get()$
    check if S_{reach}^i should be updated based on W_φ^j
    if S_{reach}^i *has been updated* **then**
        compute the updated winning region W_φ^i
        foreach *adjacent model* $\mathcal{M}^k$ *of* $\mathcal{M}^i$ **do**
            $queue.put(\langle k, i\rangle)$
return $\bigcup_{i=1}^N W_\varphi^i$

D. Correctness

Lemma 1. *Given a POMDP model $\mathcal{M}$ whose decomposition yields a set of factored models $\{\mathcal{M}^i\}_{i=1}^N$, and an almost-sure reach-avoid specification φ, the output of Algorithm 1 forms a productive winning region of $\mathcal{M}$ with respect to φ.*

Proof. Let $W := \bigcup_{i=1}^N W_\varphi^i$ denote the output of Algorithm 1. We need to show that, from every belief support in W, there exists a finite path to stay within the region W and reach some states in the set REACH of φ, according to the definition of productive winning region (cf. Section III-B).

Consider a belief support $Supp(b_0) \in W$ that belongs to the winning region W_φ^i of model $\mathcal{M}^i$. Since W_φ^i is productive towards the set S_{reach}^i by construction, there exists a finite path from $Supp(b_0)$ to $Supp(b_k) \subseteq S_{reach}^i$ while staying within W_φ^i along the path. If model $\mathcal{M}^i$ was initialized with a non-empty set $S_{reach}^i = \mathsf{REACH} \cap S^i$, then the path has reached the set REACH. Otherwise, S_{reach}^i was initialized as an empty set but updated with some winning belief support of adjacent model $\mathcal{M}^j$. Thus, $Supp(b_k) \in W_\varphi^j$ and there exists a finite path from $Supp(b_k)$ to $Supp(b_n) \subseteq S_{reach}^j$ while staying within W_φ^j along the path. We can prove by induction that the path would eventually reach some states in REACH while staying within the region W. ■

Based on the above lemma, we can follow the correctness argument of centralized shielding in Section V to show that integrating the POMCP algorithm with factored shielding via prior pruning or on-the-fly backtracking yields safe POMDP policies that satisfy almost-sure reach-avoid specifications.

Remark. We note that the union of factored winning regions computed by Algorithm 1 could be a subset of the winning region used in centralized shielding.

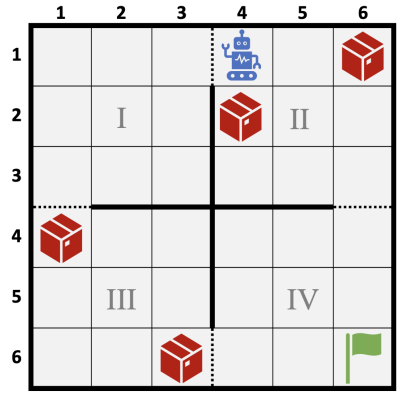

Fig. 2. A grid world example for the comparison between centralized shielding and factor shielding.

Consider the grid world environment shown in Figure 2, where the robot navigation model follows the one for the motivating example (see Section IV). When applying the centralized shielding method, we find that g_{14} is a winning belief support state, since there exists a winning policy that yields the following trajectory: $g_{14} \xrightarrow{west} g_{13} \xrightarrow{east} g_{15} \xrightarrow{south} g_{25} \xrightarrow{east} g_{26} \xrightarrow{south} g_{46} \xrightarrow{south} g_{66}$.

Now we apply Algorithm 1 for the factored shielding. We start by computing the winning region W_φ^4 of model $\mathcal{M}^4$ corresponding to room IV. We update the reach set of model $\mathcal{M}^2$ for the adjacent room II to $S_{reach}^2 = \{g_{46}, g_{56}\}$ and compute the winning region W_φ^2. We find that $g_{14} \notin W_\varphi^2$, because there does not exist a safe path for the robot to reach S_{reach}^2 from g_{14} while avoiding obstacles with probability one. The robot would collide with the obstacle in g_{16} with probability 0.2 if it moved east, or collide with the obstacle in g_{24} with probability 0.8 if it moved south. Next, we update the reach set of model $\mathcal{M}^1$ for room I to $S_{reach}^1 = \{g_{15}\}$ and compute the winning region W_φ^1. To make g_{14} a winning belief support state, we would want g_{12} and g_{13} to be contained in W_φ^1 such that the robot can move west from g_{14}. Unfortunately, starting from g_{12} or g_{13}, the robot is not guaranteed to reach $S_{reach}^1 = \{g_{15}\}$ with probability one. Thus, g_{14} is not contained the factored winning regions.

VII. Experiments

We built a prototype implementation of our shielding-enabled POMCP techniques as an extension of the PRISM model checking tool [20], with shields generated by the implementation from [13]. We then evaluated it on a set of benchmark POMDP domains adapted from [13].

- **Obstacle:** A robot navigates in $N \times N$ grid world environments, aiming to reach certain target locations while avoiding static obstacles. The reward function employed is identical to the one presented in the motivating example (cf. Section IV).
- **Refuel:** A robot navigates in $N \times N$ grid world environments and consumes energy at every movement. The robot may recharge at refuel stations to the full battery capacity E. Noisy sensors introduce uncertainty

Case Study					No Shield			Centralized Shield						Factored Shield					
								Prior			On-the-fly			Prior			On-the-fly		
Domain	Para.	$\|S\|$	$\|O\|$	$\|T\|$	Time(s)	Return	Unsafe	Time(s)	Return	Unsafe	Time(s)	Return	Unsafe	Time(s)	Return	Unsafe	Time(s)	Return	Unsafe
Obstacle (N)	6	37	20	204	0.09	978.9	1.0	0.12	929.9	0	0.20	968.1	0	0.12	921.7	0	0.16	968.7	0
	8	65	20	396	0.14	972.0	2.0	0.16	977.1	0	0.19	980.1	0	0.16	962.9	0	0.21	979.8	0
	9	82	39	464	0.13	974.0	1.9	0.18	944.0	0	0.32	962.0	0	0.18	939.3	0	0.28	964.0	0
Refuel (N, E)	6, 8	272	74	1,081	0.84	861.4	20.4	0.14	795.9	0	0.53	939.9	0	0.14	912.8	0	0.53	917.8	0
	9, 6	470	151	1,848	1.19	741.9	40.0	1.36	-261.6	0	0.81	904.5	0	1.41	-259.6	0	0.81	760.8	0
	12, 8	1,081	180	5,003	1.54	-199.0	192.1	-	-	-	-	-	-	1.15	-248.4	0	0.62	933.6	0
Rocks (N, R)	6, 3	4,157	596	4.3e**4**	0.16	1,001.1	0.7	0.33	492.6	0	0.36	1,020.9	0	0.32	840.6	0	0.32	1,062.3	0
	8, 4	3.7e**4**	2,036	4.7e**5**	0.54	1,013.3	0.5	-	-	-	-	-	-	0.99	406.6	0	0.73	1,091.5	0
	9, 6	1.2e**6**	3.3e**4**	1.8e**7**	0.87	1,008.0	1.9	-	-	-	-	-	-	1.33	-119.0	0	1.40	540.4	0

about the robot's location and battery level. The robot's goal is to reach destinations while avoiding obstacles or running out of the battery. The cost structure is: moving incurs a cost of 1, idling also costs 1 due to the lack of energy to move, and refueling to full energy is set at a cost of 3 to discourage unnecessary refueling. The reward for reaching the target is set at 1000.

- **Rocksample:** A robot navigates in $N \times N$ grid world environments with R rocks that are either valuable or dangerous to collect. To find out the quality of a rock with certainty, the robot has to sample it from an adjacent grid. The robot aims to reach target locations while avoiding sampling any dangerous rock. The cost structure assigns a value of 1 to each action, whether moving, rock-sensing, or rock-sampling, and imposes a higher cost of 20 for encountering a bad rock. Successfully reaching the target yields a reward of 1000.

We use the following hyper-parameters for the POMCP algorithm: the number of simulations in each step's online planning is 40,000; the simulation depth is 200; and the number of particles sampled from the initial state distribution is 10,000. We set the time-out for computing a winning region to be one hour. All experiments were run on a CentOS-7 machine with a 64GB Java memory limit.

Table VII shows the experimental results. For each POMDP model, we report the model parameters, the number of states $|S|$, observations $|O|$, and transitions $|T|$. We compare the performance of the baseline (i.e., POMCP without shielding) and four variants of the proposed shielding methods in terms of the following metrics: search time per planning step, expected return, and occurrences of unsafe states. The results shown in Table VII are the average over 10 runs of each method. We draw the following key insights from the results.

The proposed shielding methods can guarantee safety but the baseline does not. Across all models, the POMCP algorithm without shielding yields non-zero occurrences of unsafe states, while all four variants of the proposed shielding methods avoid unsafe states completely.

The proposed shielding methods have comparable search time per planning step with the baseline. This means that imposing shielding adds negligible overhead for the online planning. In some cases (e.g., Refuel(6,8)), shielding methods yield faster search than the baseline thanks to the pruning of tree branches. It only takes a few seconds to pre-compute winning regions for the shielding in most cases (except those time-out cases mentioned below).

Factored shielding has better scalability than centralized shielding. For some cases including Refuel(12,8), Rocksample(8,4), and Rocksample(9,6), centralized shielding fails to compute (fixpoint) winning regions before the time-out. By contrast, factored shielding scales up to large POMDP models with millions of states, since it only requires us to compute a set of smaller factored winning regions.

On-the-fly backtracking generally yields higher expected return than prior pruning. Intuitively, on-the-fly backtracking chooses optimal actions based on node values estimated from safe simulated paths, while prior pruning only considers safety at the root node level and may choose locally optimal actions that cost more in the long run.

VIII. Conclusion

In this work, we developed four distinct shielding methods, differing in how the shields are computed and integrated with the POMCP algorithm, for safe POMDP online planning with respect to almost-sure reach-avoid specifications. Experimental results on a set of benchmark domains demonstrate that the proposed shielding methods successfully guarantee safety (unlike the baseline POMCP without shielding), with negligible impact on the runtime for online planning. In particular, factored shielding methods can scale up to solve large POMDP models with millions of states.

There are several directions to explore for possible future work. First, we will evaluate the proposed methods on a wider range of POMDP domains, beyond those benchmark domains considered in our experiments. Second, we will explore a principled way to achieve efficient POMDP model decomposition schemes for factored shielding. Finally, we would like to apply the proposed methods to robotic tasks in real-world scenarios (e.g., autonomous driving).

Acknowledgments

This work was supported in part by U.S. National Science Foundation under grant CCF-1942836, U.S. Office of Naval Research under grant N00014-18-1-2829, U.S. Air Force Office of Scientific Research under grant FA9550-21-1-0164 and the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the grant sponsors.

References

[1] S. Thrun, *Probabilistic robotics*. MIT Press, 2006.

[2] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, 2022.

[3] S. Koenig, R. Simmons, *et al.*, "Xavier: A robot navigation architecture based on partially observable markov decision process models," *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, no. partially, pp. 91–122, 1998.

[4] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *2015 Ieee International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 454–460.

[5] S. Sheng, E. Pakdamanian, K. Han, Z. Wang, J. Lenneman, D. Parker, and L. Feng, "Planning for automated vehicles with human trust," *ACM Transactions on Cyber-Physical Systems*, vol. 6, no. 4, pp. 1–21, 2022.

[6] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu, "Searching and tracking people with cooperative mobile robots," *Autonomous Robots*, vol. 42, no. 4, pp. 739–759, 2018.

[7] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[8] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs," *Advances in neural information processing systems*, vol. 23, 2010.

[9] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online POMDP planning with regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[10] J. Lee, G.-H. Kim, P. Poupart, and K.-E. Kim, "Monte-carlo tree search for constrained POMDPs," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[11] M. Khonji, A. Jasour, and B. C. Williams, "Approximability of constant-horizon constrained POMDP," in *IJCAI*, 2019, pp. 5583–5590.

[12] Y. Wang, A. A. R. Newaz, J. D. Hernández, S. Chaudhuri, and L. E. Kavraki, "Online partial conditional plan synthesis for POMDPs with safe-reachability objectives: Methods and experiments," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 932–945, 2021.

[13] S. Junges, N. Jansen, and S. A. Seshia, "Enforcing almost-sure reachability in POMDPs," in *International Conference on Computer Aided Verification*. Springer, 2021, pp. 602–625.

[14] S. Carr, N. Jansen, S. Junges, and U. Topcu, "Safe reinforcement learning via shielding under partial observability," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 14 748–14 756.

[15] G. Mazzi, A. Castellini, and A. Farinelli, "Risk-aware shielding of partially observable monte carlo planning policies," *Artificial Intelligence*, vol. 324, p. 103987, 2023.

[16] A. Undurti and J. P. How, "An online algorithm for constrained POMDPs," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3966–3973.

[17] K. Chatterjee, A. Elgyütt, P. Novotny, and O. Rouillé, "Expectation optimization with probabilistic guarantees in POMDPs with discounted-sum objectives," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 4692–4699.

[18] K. Chatterjee, P. Novotnỳ, G. Pérez, J.-F. Raskin, and Đ. Žikelić, "Optimizing expectation with guarantees in POMDPs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[19] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[20] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.