Point-Based Methods for Model Checking in Partially Observable Markov Decision Processes

Maxime Bouton
boutonm@stanford.edu
Stanford University
Stanford, CA

Jana Tumova
tumova@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Mykel J. Kochenderfer
mykel@stanford.edu
Stanford University
Stanford, CA

Abstract

Autonomous systems are often required to operate in partially observable environments. They must reliably execute a specified objective even with incomplete information about the state of the environment. We propose a methodology to synthesize policies that satisfy a linear temporal logic formula in a partially observable Markov decision process (POMDP). By formulating a planning problem, we show how to use point-based value iteration methods to efficiently approximate the maximum probability of satisfying a desired logical formula and compute the associated belief state policy. We demonstrate that our method scales to large POMDP domains and provides strong bounds on the performance of the resulting policy.

Introduction

Designing decision making strategies for robotic systems in uncertain environments can be challenging. In many applications, the agent is equipped with sensors that are not capable of detecting all the relevant features of the environments. Sensors may not be able to detect objects through walls or directly measure the intentions of humans. Algorithms must generate strategies that are both efficient and reliable even in situations where all the information about the environment is not accessible. In addition, the resulting policies must exhibit strong guarantees on their performance.

A principled way to take into account both stochastic dynamics and state uncertainty is to model the environment as a partially observable Markov decision process (POMDP). The objective is often specified using a reward function. The agent seeks to find a strategy that maximizes the expected accumulated reward over time. Defining reward functions can be very challenging and can lead to a value alignment problem, where the agent does not behave as expected (Hadfield-Menell et al. 2017). Although existing planning algorithms can generate approximately optimal policies, it may not be straightforward how to interpret the performance of the policy through expected accumulated rewards.

In this work, we focus on the problem of synthesizing policies that achieve a desired objective expressed by a logical formula in a POMDP. We consider linear temporal logic (LTL) (Pnueli 1977) as the framework for specifying the objective. We are interested in computing the probability of satisfying the desired formula when following the resulting policy. This problem is known as *quantitative* model checking (Baier and Katoen 2008). In general, the problem of computing a policy that has the best probability of satisfying a logical formula in a POMDP is undecidable (Chatterjee, Chmelik, and Tracol 2013). However, it is possible to derive approximate solutions to the problem with confidence bounds (Hauskrecht 2000).

We propose a methodology to approximately solve quantitative model checking problems in POMDPs. We show that the problem of finding a policy maximizing the satisfaction of the objective can be formulated as a reward maximization problem. This consideration allows us to benefit from efficient approximate POMDP solvers, such as SARSOP (Kurniawati, Hsu, and Lee 2008), to solve the original model checking problem. In addition, the bounds provided by the solver constitute strong guarantees on the performance of the resulting policy. We apply our methodology to classical POMDP domains and demonstrate that it can scale to larger environments than previous methods. We empirically verify that the probability of success of the policy is consistent with the upper and lower bounds provided by the solver. Finally, we compare the performance of point-based methods against previous work (Norman, Parker, and Zou 2017).

Related Work

Model checking in finite state Markov decision processes (MDPs) has been studied extensively and relies on two main solving strategies: value iteration and linear programs (Baier and Katoen 2008; Lahijanian, Andersson, and Belta 2011). These algorithms scale polynomially in the size of the MDP and efficient tools for probabilistic model checking can synthesize policies satisfying an LTL formula in MDPs with several millions states (Kwiatkowska, Norman, and Parker 2011; Dehnert et al. 2017). However, these tools have little support for environments where the state is not observable, and current methods cannot scale to large POMDPs useful for robotics applications.

The general problem of finding a policy satisfying an LTL formula in an infinite horizon POMDP is undecidable (Chat-

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

terjee, Chmelik, and Tracol 2013). However, one can often compute approximate solutions by relaxing some aspects of the problem. A possible approach consists of restricting the space of policies to finite state controllers. This assumption can significantly reduce the search space. Chatterjee et al. propose an exact algorithm relying on some heuristics to find policies satisfying a formula with probability 1. This algorithm has been used to synthesize policies in a drone surveillance problem (Svorenová et al. 2015). Other algorithms solve the quantitative model checking problem using parameter synthesis (Junges et al. 2018) or a variant of value iteration (Sharan and Burdick 2014). The restriction to classes of policies with a limited number of internal states allows those approaches to scale to domains with thousands of states. However, in many applications, finite state policies might not be expressive enough to solve the problem. Instead, the policy must be represented as a mapping from a belief state (a distribution over states) to an action.

Norman, Parker, and Zou addresses the problem of belief state planning with LTL specifications by discretizing the belief space and formulating an MDP over this space (Norman, Parker, and Zou 2017). In problems where the state space has more than a few dimensions, discretizing the belief space becomes intractable. We demonstrate that our method scales to problems with an order of magnitude more hidden states. Similarly, abstraction refinement methods were proposed to discretize the belief space in linear Gaussian POMDPs (Haesaert et al. 2018). Another approach for control in the belief space with LTL specifications linear Gaussian systems uses sampling based methods (Vasile et al. 2016). Wang, Chaudhuri, and Kavraki proposed an online search method to only explore belief points reachable from the current belief but their approach is limited to safe reachability objectives where the agent maximizes the probability of reaching a goal state while avoiding dangerous states (Wang, Chaudhuri, and Kavraki 2018). Alternative methods can check that a given belief state policy satisfies a safety or optimality criterion using barrier certificates but do not allow for policy synthesis (Ahmadi et al. 2018).

In this work, we propose a method to synthesize policies mapping belief states to actions with an LTL specification in a POMDP. We show that we can benefit from the advances in POMDP planning algorithms to solve model checking problems efficiently and avoid a naive discretization of the belief space. In contrast with previous work, we do not assume that the labels constituting the LTL formula are observable. In addition, our method handles stochastic observation models.

Background

This section reviews partially observable Markov decision processes and linear temporal logic.

Partially Observable Markov Decision Processes

Sequential decision making problems with state uncertainty can be modeled as partially observable Markov decision processes (POMDPs). They are formally defined by the tuple $(\mathscr{S}, \mathscr{A}, \mathscr{O}, T, O, R, \gamma)$ where $\mathscr{S}$ is a finite state space, $\mathscr{A}$ a finite action space, $\mathscr{O}$ a finite observation space, T a transition model, O an observation model, R a reward function, and γ a discount factor. The transition model describes the probability of transitioning to a state s' when taking an action $a \in \mathscr{A}$ in a state s: $T(s' \mid s,a) = \Pr(s' \mid s,a)$. When executing an action a in a state s, the agent receives a scalar reward given by the function $R(s,a)$. The observation model represents the probability of observing $o \in \mathscr{O}$ while having executed action a and being in state s': $O(o \mid s',a) = \Pr(o \mid s',a)$.

During the decision process, the agent cannot sense the true state of the environment. Instead it maintains a belief that reflects its internal knowledge of the state. The *belief state* is a probability distribution over all possible states, $b : \mathscr{S} \to [0,1]$, and $b(s)$ represents the probability of being in state s. In POMDPs with finite states, actions, and observations, the belief b is updated after taking action a and observing o using the following equation:

$$b'(s') \propto O(o \mid s',a) \sum_s T(s' \mid a,s) b(s) \tag{1}$$

A policy is a mapping from beliefs to actions. Given a policy π, an induced trajectory is a trajectory generated by an agent following π from a given belief point. The solution to a POMDP is a policy π^* that, if followed, maximizes the expected discounted sum of immediate rewards. The optimal policy can be extracted from the optimal belief action utility function $U^*(b,a)$ as follows:

$$\pi^*(b) = \arg\max_a U^*(b,a) \tag{2}$$

where $U^*(b,a)$ represent the accumulated discounted reward obtained when following the optimal policy after taking action a in belief b. We note $U^*(b) = \max_a U^*(b,a)$ the belief state utility function (also called value function).

When performing model checking, a convenient approach is to label the states of the POMDP and express the property we wish to verify in terms of these labels. The labels are atomic propositions that evaluate to true or false at a given state. We augment the definition of a POMDP with a finite set of atomic propositions Π, and L a mapping, $L : \mathscr{S} \to 2^{\Pi}$, giving the set of atomic propositions satisfied at a given state. We do not assume that the labels are observable. The agent should infer the labels from the observations.

In this work, we focus on POMDPs with finite states, actions, and observations. We discuss possible extensions to continuous spaces in the conclusion.

Linear Temporal Logic

Linear Temporal Logic (LTL) is an extension to propositional logic with temporal operators. An LTL formula is built of atomic propositions according to the following grammar:

$$\phi ::= p \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \mathsf{G}\phi \mid \mathsf{F}\phi \mid \phi_1 \mathsf{U} \phi_2 \mid \mathsf{X}\phi \tag{3}$$

where p is an atomic proposition, ϕ, ϕ_1, and ϕ_2 are LTL formulas, $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) are logical operators, and G (globally), F (eventually), U (until), and X (next) are temporal operators (Baier and Katoen 2008). In this work we use LTL as a language to specify the objective of the problem. For example, safe-reachability objectives: "avoid state A and reach state B" are specified

by the formula $\neg A \mathsf{U} B$, persistent tasks: "keep visiting A" are represented by the formula $\mathsf{GF}A$.

The satisfaction of an LTL formula is evaluated on an infinitely long trajectory in the environment. A labelling function maps each state of the environment to the set of atomic propositions holding in that state. The satisfaction of the formula can be verified by analyzing the sequence of atomic propositions generated by a trajectory. Even if the trajectory is continuous in time, the sequence of atomic propositions needs to be discrete.

Proposed Approach

This section presents our approach to solve the quantitative model checking problem using a POMDP formulation. We first demonstrate how to formulate a planning problem from a given model checking problem. Then, we explain how to approximately compute a policy that maximizes the probability of satisfying a given LTL formula. Finally we discuss how the convergence error of the solver can be used as a confidence interval on the resulting performance.

Problem Formulation

The problem of interest consists of computing the maximum probability of satisfying a given linear temporal logic formula ϕ when starting in an initial belief point b in a POMDP.

Given a policy π, $\Pr^{\pi}(b \models \phi)$ represents the probability that a trajectory induced by π starting from belief b will satisfy the LTL formula ϕ. The quantity we wish to compute is expressed as follows:

$$\Pr^{\max}(b \models \phi) = \max_{\pi} \Pr^{\pi}(b \models \phi) \tag{4}$$

Such problem is referred to as *quantitative* model checking as opposed to *qualitative* model checking, which consists of finding a policy satisfying the formula with probability 1 (Chatterjee et al. 2015). In this work, the atomic propositions forming the LTL formula are defined over the states of the POMDP. Hence, the value of the atomic propositions is not observed by the agent. Instead, we will show that our formulation captures this information in the belief state.

Reachability Problems

Point-based value iteration methods can scale to POMDPs with many thousands states (Kurniawati, Hsu, and Lee 2008; Shani, Pineau, and Kaplow 2013). Those solvers have been designed to solve reward maximization problems. We explain how to formulate reachability problems as reward maximization problems so we can use these solvers.

A reachability problem consists of computing the maximum probability of reaching a given set of states. If B is a propositional formula then the reachability problem corresponds to computing $\Pr^{\max}(b \models \mathsf{F}B)$. For simplicity of the notation, we will also denote B, the set of states where the propositional formula expressed by B holds true. A reachability problem can be interpreted as a planning problem where the goal is to reach the set B. This problem is addressed by defining the following reward function:

$$R_{\text{Reachability}}(s,a) = \begin{cases} 1 & \text{if } s \in B \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In addition, the states in the set B are made terminal states and the initial value of $\Pr^{\max}(b \mid \mathsf{F}B)$ is initialized to 0 for any belief states. We can interpret the reachability problem as a reward maximization problem as follows:

$$\Pr^{\max}(b \models \mathsf{F}B) = \max_{\pi} \mathbb{E}[\sum_{t}^{\infty} R_{\text{Reachability}}(s_t, \pi(b_t)) \mid s_0 \sim b] \tag{6}$$

The right side of this equation corresponds to solving a POMDP planning problem with a value based method (Kochenderfer 2015). The maximization is over the policy space. In a POMDP, policies map belief states to actions rather than states to actions. The search problem becomes much harder than in MDPs and the value iteration algorithm can no longer scale. It has been proven that computing the maximum expected reward in a POMDP is undecidable (Madani, Hanks, and Condon 1999). Instead, we will rely on approximate methods that scales to POMDP domains with tens of thousands of states. This step is discussed in depth in the section on approximate solution techniques. The next section discusses the generalization to any LTL formula.

From LTL Satisfaction to Reachability

Product POMDPs In this step, we define a new POMDP such that solving the original quantitative model checking problem reduces to a reachability problem in this model.

It is known that any LTL formula can be represented by a deterministic Rabin automaton (Baier and Katoen 2008), which can be defined as follows:

Deterministic Rabin Automata (DRA): A deterministic Rabin automaton is a tuple $\mathscr{R} = (Q, \Pi, \delta, q_0, F)$ where Q is a set of states, Π a set of atomic propositions, $\delta : Q \times 2^{\Pi} \to Q$ is a transition function, q_0 is an initial state, and F is an acceptance condition: $F = \{(L_1, K_1), \ldots, (L_k, K_k)\}$ where L_i and K_i are sets of states for all i.

A trajectory of a Rabin automaton is an infinite sequence of states $\tau = q_0 q_1 \ldots$, where $q_{i+1} = \delta(q_i, \sigma)$ for an input $\sigma \in 2^{\Pi}$. We say that a trajectory is accepting if there exists i such that: $\inf(\tau) \cap K_i \neq \emptyset$ and $\inf(\tau) \cap L_i = \emptyset$ where $\inf(\tau)$ is the set of states visited infinitely often in the trajectory. By converting the LTL formula into a DRA, we have a direct equivalence between accepting trajectories and trajectories satisfying the formula.

In general, converting an LTL formula into a DRA results in a finite state machine with a number of states double exponential in the number of atomic propositions in the formula. In practice, a lot of heuristics can be used to reduce the number of states in the automaton to a reasonable number. We give an example of the automaton resulting from converting $\mathsf{G}\neg A \wedge \mathsf{F}B$ in Fig. 1.

Product POMDP: For a POMDP $\mathscr{P}$, and DRA $\mathscr{R}$, we define a product $\mathscr{P} \otimes \mathscr{R}$ as a POMDP: $\mathscr{P}' = (\mathscr{S} \times Q, \mathscr{A}, \mathscr{O}, T', O, L)$ where the state space is the Cartesian product of the state space of $\mathscr{P}$ and $\mathscr{R}$ and the transition function satisifies:

$$T'((s,q),a,(s',q')) = \begin{cases} T(s,a,s') & \text{if } q' = \delta(q, L(s)) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

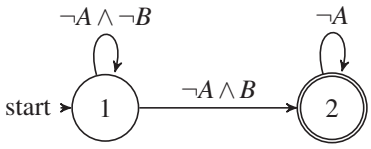

Figure 1: Illustration of an automaton generated by converting the LTL formula $\mathsf{G}\neg A \wedge \mathsf{F}B$. State 2 must be visited infinitely often to satisfy the formula. Each propositional formula on the edges represents possibly multiple transitions labeled with the subsets of atomic propositions that satisfy the formula on the edge.

all the other elements of the product are the same as in the original POMDP. In the product, some transitions are prevented by the automaton. We can notice that the transition function defined is no longer a probability distribution. In practice, we can add an additional sink state such that if $\delta(q, L(s)) = \emptyset$, the system transitions in the sink state with probability 1. The new transition function ensures that trajectories that end up in the sink state are not accepted by the automaton (they are violating the specification).

Let aside the model checking problem, the construction of the product POMDP can be interpreted as a principled way to augment the state space in order to account for temporal objective. In addition, one can note that this state space extension is not always necessary. For formulas involving only a single until (U) or eventually (F) temporal operators, the problem can be directly expressed as a reachability problem and does not require a state space augmentation.

Maximal End Components The next step consists of identifying a set of states B in the product POMDP, such that reaching a state in this set guarantees the satisfaction of the formula. We call those states *success states*.

From the definition of the DRA, we find that an infinitely long trajectory satisfying the formula must visit certain states infinitely often and others only finitely often. We first start to compute the sets of states that are visited infinitely often in the product POMDP, that is the maximal end component of a POMDP. More precisely, we need to find the maximal end components of the underlying MDP defined by $(\mathcal{S} \times \mathcal{Q}, \mathcal{A}, T')$. Starting from any state, with any policy, the agent will end up in a maximal end component if we consider infinitely long trajectories. Maximal end components can be computed by a graph algorithm that scales polynomially with the size of the state space (Baier and Katoen 2008). Once the end components have been found, we must identify the success states.

Success States: (Baier and Katoen 2008) Given a product POMDP $\mathcal{P}'$, its underlying MDP is noted $\mathcal{M}'$. A state contained in a maximal end component EC of $\mathcal{M}'$ is a success state if there exists an i such that $K_i \in EC$ and $L_i \notin EC$, where K_i and L_i results from the accepting conditions of the DRA used to form the product POMDP.

From the previous definition, we can conclude that from a success state, there is a probability of 1 of satisfying the LTL formula associated with the Rabin automaton. We can define a reachability reward function associated to the set of success states and compute the probability of success at a given belief point using Eq. (6).

The first steps of the model checking approach (product POMDP and reduction to reachability) are identical for POMDPs and MDPs. They are independent of the structure of the observation space and are agnostic to partial observability. State uncertainty will play a role in the last step, which consists of solving the reachability problem.

Theorem: Given a POMDP and an LTL formula ϕ, the optimal value function of the product POMDP with the reachability reward function associated with the set of success states satisfies: $U^*(b) = \Pr^{\max}(b \models \phi)$, where b is a belief state in the product POMDP. In addition, there is a one to one mapping between the policy maximizing the value function in the product POMDP and the policy maximizing $\Pr(b \models \phi)$.

Proof Sketch: The construction of the product POMDP, and the definition of success states give the following:

$$\Pr^{\max}_{\mathcal{P}}(b \models \phi) = \Pr^{\max}_{\mathcal{P}'}(b \models \mathsf{F}B) \tag{8}$$

where on both sides, b is a belief of the product states, that is a belief over both the state of $\mathcal{P}$ and the state of the DRA associated with ϕ, and B is the set of success states in $\mathcal{P}'$. When updating the belief using Eq. (1), the transition model from the product POMDP is used. Finally, Eq. (6) holds from the construction of the reachability reward function and the definition of the belief state value function of a POMDP. More precisely, Eq. (6) can be proven by formulating a belief state MDP (Kochenderfer 2015) and use the equivalent result for MDPs (Baier and Katoen 2008).

The agent cannot observe whether it has reached an end component or not, but the belief state characterizes the confidence on whether or not it is in an end component. Previous works often assume that the end components are observed, our algorithm allows to relax this assumption by maintaining a belief on both the state of the environment and the state of the automaton.

Approximate Solution Techniques

The previous sections illustrated how to convert the quantitative model checking problem into a reward maximization problem. This section describes how to solve this problem using existing POMDP planning algorithms and how to interpret the convergence bounds with respect to the problem of interest. As we have shown, $\Pr^{\max}(b \models \phi)$ can be interpreted as a belief value function for a specific POMDP. This section discusses how to compute such value function.

Solving POMDPs exactly is generally intractable (Kochenderfer 2015; Madani, Hanks, and Condon 1999), however approximation techniques have been developed. Approximation methods rely on restricting the policy space, either by considering finite-state controllers or alpha vector representations. Previous work addressed the problem of finding finite state controllers (Junges et al. 2018; Chatterjee et al. 2015). This paper focuses on alpha vector representations of the policy and the value function. Alpha vectors can be used to represent both the policy and the value function. Hence, we can approximate the quantitative model checking problem and not only the policy synthesis problem.

Alpha vectors are $|\mathscr{S}|$-dimensional vectors defining a linear function over the belief space. Given a set of alpha vectors $\Gamma = \{\alpha_1, \ldots, \alpha_n\}$, the value function is defined as follows: $U(b) = \max_{\alpha \in \Gamma} \alpha^\top b$ Point based Value Iteration (PBVI) algorithms are a family of POMDP solvers that involves applying a Bellman backup to a set of alpha vectors in order to approximate the optimal value function. Shani, Pineau, and Kaplow survey various PBVI methods. In this work, we used SARSOP (Kurniawati, Hsu, and Lee 2008), which has shown state-of-the-art performance in terms of scalability. PBVI algorithms sample the belief space and compute an alpha vector associated to each belief point to approximate the value function at that point. SARSOP differs from other PBVI algorithms by relying on a tree search to explore the belief space. It maintains an upper and lower bound on the value function, which are used to guide the search close to optimal trajectories. The algorithm is given an initial belief point and only explores relevant regions of the belief space. That is, regions that can be reached from the initial belief point under optimality conditions.

PBVI algorithms, often offer convergence guarantees specified in upper and lower bound on the value function. A precision parameter ε is provided and control the tightness of the convergence (by controlling the depth of the tree in SARSOP for example) which yields to:

$$|\overline{U^*(b_0)} - \underline{U^*(b_0)}| < \varepsilon \tag{9}$$

Given a formula ϕ, we have show how to build a product POMDP in which we have the equivalence between the value function $U^*(b)$ and $\Pr^{\max}(b \models \phi)$. As a consequence, for a given precision parameter, we can directly translate the bounds on the value function in the product POMDP in terms of probability of success for our problem of quantitative model checking:

$$|\overline{\Pr^{\max}(b_0 \models \phi)} - \underline{\Pr^{\max}(b_0 \models \phi)}| < \varepsilon \tag{10}$$

where $\overline{\Pr^{\max}(b_0 \models \phi)}$ is an upper bound over the actual probability of satisfaction, $\underline{\Pr^{\max}(b_0 \models \phi)}$ is a lower bound, and b_0 is the initial belief. With an infinite computation time, an arbitrary ε can be reached. However in practice only a minimum ε can be achieved within the computation budget. The original implementation of SARSOP relies on a discount factor. In this work, the discount factor is set to one such that the obtained value function matches exactly with the probability of satisfaction of the LTL formula.

The proposed methodology to solve quantitative model checking problems in POMDPs is agnostic to the planning algorithm. Although we focused the discussion on PBVI solvers, any belief state planner could be used. The strength of the guarantees are directly dependent on the choice of the underlying planning algorithm. For example, one could use the QMDP or FIB approximations to only compute an upper bound on the probability of success (Hauskrecht 2000). Our implementation allows the user to easily choose the underlying algorithm among the one available in POMDPs.jl (Egorov et al. 2017) a POMDP planning library.

Experiments

We evaluate our methodology on three discrete POMDP domains from the literature. The first one is a partially observable slippery grid world, the second one is the rock sample problem (Smith and Simmons 2004), and the third is a drone surveillance problem (Svorenová et al. 2015). Those domains have a grid world like structure and can easily be scaled to different size of state and observation spaces to evaluate the scalability of our approach. More details can be found in the source code and in the supplementary material.

Partially Observable Grid World This domain is an $n \times n$ grid with three labels: A, B, and C associated to some cells in the grid. The agent can choose to move left, right, up, and down. It reaches the desired cell with a probability of 0.7 and moves to another neighboring cell with equal probability otherwise. The agent receives a noisy observation of its position generated from a uniform distribution over the neighboring cells (vanish for distances greater than 1). The agent is initialized to a cell in the grid world with uniform probability. We investigated the following specifications:

- $\phi_1 = \neg C \mathsf{U} A \wedge \neg C \mathsf{U} B$: The agent must visit states A and B in any order while avoiding state C. This formula is a constrained reachability objective and does not require to form a product POMDP.
- $\phi_2 = \mathsf{G} \neg C$: The agent must never visit state C.

The precision of the solver is set to 1×10^{-2}.

Drone Surveillance The drone surveillance problem is inspired by Svorenová et al. (Svorenová et al. 2015). An aerial vehicle must survey regions in the corners of a grid like environment while avoiding a ground agent. The drone can observe the location of the ground agent only if it is in its field of view delimited by a 3×3 area centered at the drone location. We labeled the states as A when the drone is in the bottom left corner, B when it is in the top right corner, and `det` when it can be detected by the ground agent (when it is on top of it). We analyzed one formula: $\neg \texttt{det} \mathsf{U} B$. The drone should eventually reach region B without being detected. Note that this is already a reachability objective and does not require the construction of a product POMDP. The precision is set to 1×10^{-2}.

Rock Sample The rock sample problem models a rover exploring a planet and tasked to collect interesting rocks. The environment consists of a grid world with rocks at a known location as well as an exit area. The rocks can be either good or bad and their status is not observable. The robot can move deterministically in each direction or choose to sample a rock (when on top of it), or use its long range sensor to check the quality of a rock. The long range sensor returns the true status of a rock with a probability decaying exponentially with the distance to the rock. The problem ends when the robot reaches the exit area, this state is labeled as `exit`. In addition we defined two labels for situations when the robot pick a good rock or a bad rock respectively labeled `good` and `bad`. This paper considers three different formulas:

- $\phi_1 = \mathsf{G}\neg\mathtt{bad}$: This formula expresses that the robot should never pick up a bad rock. There exist a trivial policy that satisfies this formula which is to never pick up any rocks.
- $\phi_2 = \mathsf{F}\mathtt{good} \wedge \mathsf{F}\mathtt{exit}$: This formula expresses that the robot should eventually pick a good rock and eventually reach the exit. Since the exit is a terminal state, the robot must pick up a good rock before reaching the exit. This policy cannot be satisfied with a probability 1 since there is a possibility that all the rocks present are bad.
- $\phi_3 = \mathsf{F}\mathtt{good} \wedge \mathsf{F}\mathtt{exit} \wedge \mathsf{G}\neg\mathtt{bad}$: This formula is a combination of the two previous specifications. In addition of bringing a good rock and reaching the exit the robot must not pick a bad rock. A video demonstrating the resulting strategy is provided in the supplementary material.

For this domain, the precision of the solver is set to 1×10^{-3}.

Results

We applied the proposed methodology on different sizes of the proposed domains with different formulas. We use SARSOP as the underlying POMDP planning algorithm to solve the quantitative model checking problems. Our approach is agnostic to the choice of the planning algorithm and other methods could have been used. However, SARSOP is a good candidate for the task since it is one of the most scalable offline POMDP planners (Kurniawati, Hsu, and Lee 2008). In addition, it provides bounds on the results, which can be translated into guarantees on the probability of success.

We compared the performance of SARSOP with the algorithm used by Norman, Parker, and Zou. It consists of computing an upper bound by discretizing the belief space and performing Bellman backups on each of the belief points (Lovejoy 1991). The main drawback of this algorithm is that the belief space is high dimensional (12545 dimensions for the largest rock sample), and the size of the grid grows exponentially. Fig. 2 illustrates the benefits of using SARSOP instead of the Lovejoy algorithm. The discretization scheme is controlled by a granularity parameter m, the bigger m is, the more belief points are used. The Lovejoy line is obtained by varying m from 1 to 8, while the SARSOP line is obtained by specifying different precision targets. In the log scale figure, we can see that it takes much longer time to reach a given precision using the Lovejoy algorithm than SARSOP. In addition, we can see the exponential growth of the number of belief points. As a reference, we added the precision given by QMDP (Littman, Cassandra, and Kaelbling 1995) and FIB (Hauskrecht 2000) which are two algorithms to compute upper bound on the value of a POMDP. Point-based methods provide both an upper and desired bound and allow the user to specify the precision. Hence there is no need to use an abstraction refinement mechanism to choose the right granularity of the belief space as done in previous work (Norman, Parker, and Zou 2017).

Table 1 summarizes the performance of our approach in solving different tasks. In each case, we report the lower bound on $\Pr^{\max}(b_0 \models \phi)$ as well as the precision ε described in previous sections. The upper bound is the sum of the two. In addition, we report the solving time, it takes into account both the time to compute the maximal end components in the product POMDP as well as the time taken by SARSOP to solve the problem. The MEC column reports the time needed to identify the success states and construct the product POMDP (if needed). To control the number of iterations used by SARSOP, we used a threshold on the precision, ε *i.e.* after each iteration we check if the precision is lower than the threshold and return the policy and the probability of success if it is. The $|\Gamma|$ columns reports the number of belief points used by the point-based method.

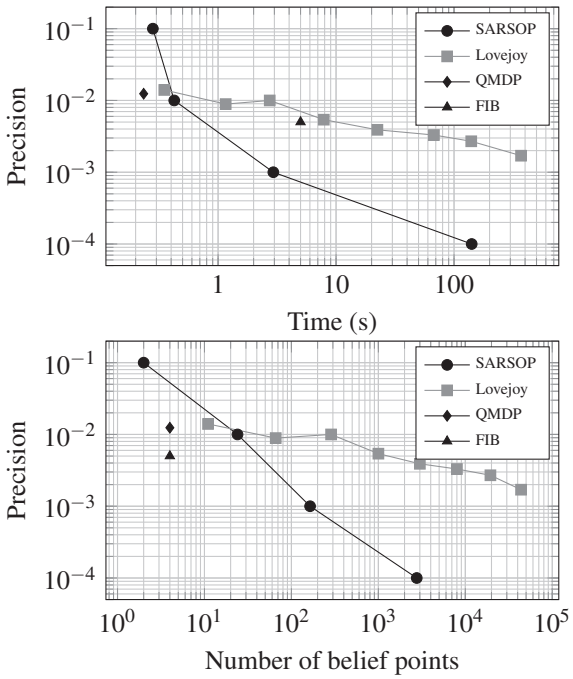

Figure 2: Illustration of the time precision trade-off for different algorithms providing upper bounds on the value function in a POMDP. Lovejoy is the algorithm used by Norman, Parker, and Zou. To compute the precision, we used the lower bound computed using SARSOP as a reference. The experiments are carried on a 3×3 partially observable grid world domain.

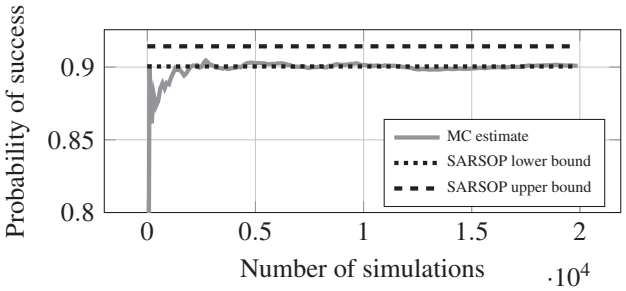

Figure 3: Estimate of the probability of success of a policy generated by SARSOP. We simulated 10000 episodes estimated the probability of success. We compare this result with the upper and lower bound provided by SARSOP.

Table 1: Performance of POMDP model checker.

Domain	$\lvert\mathcal{S}\rvert$ / $\lvert\mathcal{A}\rvert$ / $\lvert\mathcal{O}\rvert$	LB	ε	$\lvert\Gamma\rvert$	MEC (s)	Time (s)
PO Grid World						
[10, 10] ϕ_1	101 / 4 / 101	0.904	9.9×10^{-3}	3452	0.64	207.2
[10, 10] ϕ_2		0.0099	0	1	0.13	0.4
Drone Surveillance						
[5,5]	626 / 5 / 10	0.96	9×10^{-3}	4812	0.73	95.5
[5,5] (U)		0.94	8×10^{-3}	4277	0.73	78.3
[7,7] (U)	2402 / 5 / 10	0.94	1.9×10^{-2}	41799	4.8	12587.5
Rock Sample						
[4,4] ϕ_1	65 / 7 / 3	1.0	0.0	1	0.03	0.02
[4,4] ϕ_2		0.749	9.2×10^{-5}	13	0.09	0.3
[4,4] ϕ_3		0.744	2×10^{-4}	23	0.10	0.4
[5,5] ϕ_1	201 / 8 / 3	1.0	0.0	1	0.19	0.11
[5,5] ϕ_2		0.879	2.8×10^{-4}	24	0.70	0.5
[5,5] ϕ_3		0.865	9×10^{-4}	56	0.70	0.8
[7,7] ϕ_1	12545 / 13 / 3	1.0	0.0	1	11.3	13.4
[7,7] ϕ_2		0.990	9×10^{-4}	378	50.6	77.5
[7,7] ϕ_3		0.979	9×10^{-4}	301	53.5	87.2

We empirically verify the correctness of the bound provided by SARSOP by simulating the resulting policy in the partially observable grid world with the formula $\neg C \mathsf{U} A \wedge \neg C \mathsf{U} B$. Fig. 3 illustrates the convergence of the estimated probability of success with the number of simulation of the policy. The probability of success is estimated using a Monte Carlo estimator. We can see that the estimated value converges towards the lower bound provided by SARSOP (dotted line). In this particular example, the value of the probability of success is around 0.90. The gap between the upper and lower bound provided by the solver can be controlled with the precision, in expense of a longer time to solve. Fig. 3 shows that the resulting policy has an empirical performance consistent with the lower bound given by SARSOP.

Discussion

We have illustrated in the previous section that our approach scales to POMDP domains with many thousands states and supports different LTL specifications. We can see from Table 1, that the model checker is able to provide an approximate solution in a reasonable time. In contrast with previous work (Svorenová et al. 2015; Chatterjee et al. 2015), solving a quantitative model checking problem instead of a qualitative problem allows us to find a policy even in cases where satisfiability cannot be guaranteed with probability 1. Moreover, our technique scales to larger state spaces.

In a few cases, the solver returned a policy with perfect precision in a very short time. This is the case for $\mathsf{G}\neg C$ in grid world, and $\mathsf{G}\neg\texttt{bad}$ in rock sample. In those two cases, the probability of success can be directly extracted from the maximum end components. In the grid world example, the whole grid world is a maximal end component. The state space is fully connected under any policy because of the probabilistic transitions. As a consequence, there exists no trajectory that would not eventually visit the state C in an infinite time. This problem does not have any success states. In the rock sample problem, the transition is deterministic, there exist many trivial policies to not pick a bad rock. The robot can just stay idle, or reach the exit. In those two examples, computing the maximum end component and performing one iteration of SARSOP is enough to solve the model checking problem.

For the large version of the drone surveillance problem, the computation reached a maximum memory limit on the size of the policy and was not able to reach the desired precision. Although this problem is smaller than rock sample, the belief space has a much denser support. The drone maintains a belief over the location of the agent outside its field of view. This characteristic of the belief space makes this problem harder to approximate (Hsu, Lee, and Rong 2007).

The solution provided by our approach is approximate. Although it provides mathematical bounds on the performance, it is not possible to compute the solution exactly. Reaching an arbitrary precision would require exploring the full belief space and take an infinite time. As a consequence, for smaller domains, approaches like the one proposed by Chatterjee et al. might be more suitable (Chatterjee et al. 2015). However, our approach does allow us to find approximate solutions in domains that were intractable for previous belief state approaches to model checking in POMDPs. The formulation of the reward function in the product POMDP makes it a goal-oriented POMDP (Kolobov, Mausam, and Weld 2012). Our methodology would allow one to replace the POMDP planner by a goal-oriented POMDP solver. It would require extending the algorithm from Kolobov, Mausam, and Weld to POMDPs. A comparison with traditional POMDP planners would be an interesting future direction. The dead end framework could be a useful theoretical framework to analyze the convergence of the solvers in the product POMDPs.

Contrary to previous work (Norman, Parker, and Zou 2017), we do not assume that the labels are observable. The computed policy maps a belief in the product space (POMDP state and automaton state) to an action. In problems where the automaton state is observable, our approach could still be applied and leverage this mixed observability assumption. This property would certainly help improve the results on the large drone surveillance problem. It has been shown that PBVI algorithms can scale to even larger domains when part of the state is fully observable (Ong et al. 2009).

Conclusion

This paper proposed a methodology to solve quantitative model checking problems in POMDPs. Given an LTL formula and a POMDP model, our approach approximates the maximum probability of satisfying the formula as well as the corresponding belief state policy. We first convert the LTL formula into an automaton and construct a product POMDP between the automaton and the original POMDP model. By formulating a reward maximization problem, we have shown how to benefit from approximate POMDP planning algorithms to compute a solution to the model checking problem. Our method provides strong convergence bounds on the result. We have shown empirically that our approach applies to a variety of discrete POMDP domains, for different LTL formulas, and scales to larger problem than previous belief state techniques (Norman, Parker, and Zou 2017; Svorenová et al. 2015). We provide a Julia package for POMDP model checking available at https://github.com/sisl/POMDPModelChecking.jl.

The main limitation of the methodology is that it only applies to POMDPs with discrete state spaces. The two bot-

tlenecks are the computation of the maximal end components and the choice of the planning algorithms. For some LTL formula, like constrained reachability (Baier and Katoen 2008), or if one is interested in policy synthesis only, the reward maximization problem can be formulated without having to compute maximal end components (Sadigh et al. 2014). Our approach provides a flexible way to integrate LTL objectives in POMDP planning and allows to use any planning algorithm to allow a trade-off between convergence guarantees and scalability. Online POMDP planning algorithms could be used instead of PBVI methods to generate policies from an LTL objective at the price of lacking convergence guarantees.

Acknowledgment

This work was supported by the Honda Research Institute. The authors thank Sebastian Junges, Nils Jansen, and Emma Brunskill for their advice on the early stages of this work.

References

Ahmadi, M.; Cubuktepe, M.; Jansen, N.; and Topcu, U. 2018. Verification of uncertain POMDPs using barrier certificates. In *Allerton Conference on Communication, Control, and Computing*, 115–122.

Baier, C., and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Chatterjee, K.; Chmelik, M.; Gupta, R.; and Kanodia, A. 2015. Qualitative analysis of POMDPs with temporal logic specifications for robotics applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, 325–330.

Chatterjee, K.; Chmelik, M.; and Tracol, M. 2013. What is decidable about partially observable markov decision processes with omega-regular objectives. In *Computer Science Logic (CSL)*, 165–180.

Dehnert, C.; Junges, S.; Katoen, J.; and Volk, M. 2017. A storm is coming: A modern probabilistic model checker. In *International Conference on Computer-Aided Verification*, 592–600.

Egorov, M.; Sunberg, Z. N.; Balaban, E.; Wheeler, T. A.; Gupta, J. K.; and Kochenderfer, M. J. 2017. POMDPs.jl: A framework for sequential decision making under uncertainty. *Journal of Machine Learning Research* 18:26:1–26:5.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S. J.; and Dragan, A. D. 2017. Inverse reward design. In *Advances in Neural Information Processing Systems (NIPS)*, 6768–6777.

Haesaert, S.; Nilsson, P.; Vasile, C. I.; Thakker, R.; Aghamohammadi, A.; Ames, A. D.; and Murray, R. M. 2018. Temporal logic control of POMDPs via label-based stochastic simulation relations. In *IFAC Conference on Analysis and Design of Hybrid Systems, ADHS*, 271–276.

Hauskrecht, M. 2000. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research* 13:33–94.

Hsu, D.; Lee, W. S.; and Rong, N. 2007. What makes some POMDP problems easy to approximate? In *Advances in Neural Information Processing Systems (NIPS)*, 689–696.

Junges, S.; Jansen, N.; Wimmer, R.; Quatmann, T.; Winterer, L.; Katoen, J.; and Becker, B. 2018. Finite-state controllers of POMDPs using parameter synthesis. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 519–529.

Kochenderfer, M. J. 2015. *Decision Making Under Uncertainty: Theory and Application*. MIT Press.

Kolobov, A.; Mausam; and Weld, D. S. 2012. A theory of goal-oriented mdps with dead ends. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 438–447.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*.

Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer-Aided Verification*, 585–591.

Lahijanian, M.; Andersson, S.; and Belta, C. 2011. Control of markov decision processes from PCTL specifications. In *American Control Conference (ACC)*, 311–316. IEEE.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning (ICML)*, 362–370.

Lovejoy, W. S. 1991. Computationally feasible bounds for partially observed markov decision processes. *Operations Research* 39(1):162–175.

Madani, O.; Hanks, S.; and Condon, A. 1999. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI Conference on Artificial Intelligence (AAAI)*, 541–548.

Norman, G.; Parker, D.; and Zou, X. 2017. Verification and control of partially observable probabilistic systems. In *Real-Time Systems*, volume 53, 354–402.

Ong, S. C. W.; Png, S. W.; Hsu, D.; and Lee, W. S. 2009. POMDPs for robotic tasks with mixed observability. In *Robotics: Science and Systems*.

Pnueli, A. 1977. The temporal logic of programs. In *Symposium on Foundations of Computer Science*, 46–57.

Sadigh, D.; Kim, E. S.; Coogan, S.; Sastry, S. S.; and Seshia, S. A. 2014. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *IEEE Conference on Decision and Control (CDC)*, 1091–1096.

Shani, G.; Pineau, J.; and Kaplow, R. 2013. A survey of point-based POMDP solvers. *Journal of Autonomous Agents and Multi-Agent Systems* 27(1):1–51.

Sharan, R., and Burdick, J. W. 2014. Finite state control of POMDPs with LTL specifications. In *American Control Conference (ACC)*, 501–508.

Smith, T., and Simmons, R. G. 2004. Heuristic search value iteration for POMDPs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 520–527.

Svorenová, M.; Chmelik, M.; Leahy, K.; Eniser, H. F.; Chatterjee, K.; Cerná, I.; and Belta, C. 2015. Temporal logic motion planning using POMDPs with parity objectives: case study paper. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, 233–238.

Vasile, C. I.; Leahy, K.; Cristofalo, E.; Jones, A.; Schwager, M.; and Belta, C. 2016. Control in belief space with temporal logic specifications. In *IEEE Conference on Decision and Control (CDC)*, 7419–7424.

Wang, Y.; Chaudhuri, S.; and Kavraki, L. E. 2018. Bounded policy synthesis for POMDPs with safe-reachability objectives. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 238–246.