SAFETY AND EFFICIENCY IN AUTONOMOUS VEHICLES THROUGH PLANNING WITH UNCERTAINTY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Zachary Nolan Sunberg

May 2018

Abstract

Effective autonomous air and ground vehicles will need to maintain safety while accomplishing tasks efficiently in terms of time and other resources. Unfortunately, the objectives of safety and efficiency are fundamentally opposed because safety constraints prohibit some efficient actions. Moreover, the presence of uncertainty about the environment makes planning safe and efficient actions more difficult. Accurate models of this uncertainty coupled with effective planning algorithms can accomplish these goals.

The Markov decision process (MDP) is a systematic framework for modelling sequential decision problems with outcome uncertainty, and the partially observable Markov decision process (POMDP) adds the additional ability to model state uncertainty. MDPs and POMDPs are suitable models for a wide range of situations that an autonomous vehicle might face. However, obtaining the exact solution to a general POMDP is an intractable problem. This thesis considers approximate MDP and POMDP solutions and seeks to quantify their utility for autonomous vehicles. Specifically, it contains three contributions.

The first chapter analyzes the use of a certifiable safety constraint alongside approximate optimization in the context of unmanned aerial vehicle (UAV) collision avoidance. UAV collision avoidance is challenging in particular because small unmanned vehicles often do not have the performance capability or legal permission to perform conventional altitude-based conflict resolution maneuvers, so they must perform more complex horizontal conflict resolution. In order to ensure safety, aerospace systems have particularly stringent certification requirements that likely preclude approximate randomized planning techniques capable of handling uncertainty. This work evaluates the performance price that comes with using a simple certified policy and shows that, again, MDP and POMDP optimization can significantly reduce that price and improve both safety and efficiency simultaneously.

The second chapter considers the effects of modeling uncertainty in a difficult lane changing task for a self-driving car. Specifically, the research estimates the value of planning with the internal states of other human drivers such as their intentions and dispositions. While several other researchers have used internal-state-aware planning methods to interact with human drivers in desirable ways, they have not evaluated whether these methods offer a substantial quantitative improvement in performance over conventional approaches. This thesis shows that, in a simplified simulated setting, planning with internal states using a POMDP formulation can significantly improve both safety and efficiency simultaneously. Moreover, the thesis describes an experimental method for investigating other cases in which internal-state-aware planning may improve performance.

The benefits of POMDP planning can only be realized with algorithms that can handle real-world domains that are continuous and irregular. To that end, the third contribution of the thesis is a pair of new algorithms for solving POMDPs with continuous state, action, and observation spaces. These algorithms are motivated by analysis and numerical experiments that show that leading online POMDP solvers cannot handle continuous observation spaces. We prove that one of the previous solvers exhibits suboptimal behavior, and explain that the failure is due to two problems. First, the large observation space causes policy trees to become too wide and not deep enough. Second, the number of state particles used to represent beliefs collapses to one, causing overconfidence. The new algorithms, POMCPOW and PFT-DPW, handle these problems using progressive widening and weighted particle belief representations. Numerical experiments show that they are able to solve problems where previous methods fail.

A great deal of future work remains, including further mathematical analysis of the algorithms and testing with more realistic models of human behavior. But the contributions of this thesis to understanding the affects of uncertainty modeling and developing algorithms that apply to realistic problems are two vital steps on the path

to safe and efficient autonomy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Autonomous Vehicles

1.1.1 A World with Autonomous Transportation

- Better Safety
- More Efficiency
- Less wasted time and stress
- Better access to transportation

1.1.2 Current Progress

(Not sure what to say here)

1.1.3 Remaining Challenges

Technical

Legal and Ethical

Business

1.2 Decision Making Under Uncertainty

1.2.1 Uncertainty in Decision Making

Outcome Uncertainty

Model Uncertainty

State Uncertainty

1.2.2 Markov Decision Processes

The Markov decision process (MDP) is a mathematical formalism that can represent a wide range of sequential decision making problems. In an MDP, an agent takes *actions* that affect the *state* of the system and collects *rewards* based on the state and actions. The *Markov property* is a key attribute of MDPs which states that the next state depends (possibly stochastically) on the current state and action, and not on any previous states or actions.

Formally, an MDP is defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. The state space, $\mathcal{S}$, is the set of all possible states. The action space, $\mathcal{A}$, is the set of all actions available to the agent. The transition model, $\mathcal{T}$, represents likelihood of transitions, where $\mathcal{T}(s' \mid s, a)$ denotes the probability that the system will transition to state s' given that action a is taken in state s. The reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the rewards received while interacting in the environment, where $\mathcal{R}(s, a, s')$ denotes the reward for transitioning from s to s' when action a is taken. Finally, γ governs how reward is discounted in the future.

The objective in an MDP is to find a policy, $\pi : \mathcal{S} \to \mathcal{A}$, that maps each encountered state to an action and, when $a_t = \pi(s_t)$, maximizes the cumulative expected reward,

$$E\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\right] \tag{1.1}$$

where the subscript t is the time index. The state action value function, $Q(s, a)$, is defined as the expectation of the future cumulative reward given that the agent starts in state s, immediately takes action a, and then follows the optimal policy.

1.2.3 Partially Observable Markov Decision Processes

A partially observable Markov decision process (POMDP) is similar to and MDP except that the agent cannot directly observe the state. Instead, the agent only has access to observations that are generated probabilistically based on the actions and latent true states. A POMDP is defined by the 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathcal{Z}, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$, and γ have the same meaning as in an MDP. Additionally, $\mathcal{O}$, is the observation space, and $\mathcal{Z}$ is the observation model. $\mathcal{Z}(o \mid s, a, s')$ is the probability or probability density of receiving observation o in state s' given that the previous state and action were s and a.

Information about the state may be inferred from the entire history of previous actions and observations and the initial information, b_0. Thus, in a POMDP, the agent's policy is a function mapping each possible history, $h_t = (b_0, a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t)$ to an action. In some cases, each state's probability can be calculated based on the history. This distribution is known as a *belief*, with $b_t(s)$ denoting the probability of state s.

The belief is a sufficient statistic for optimal decision making. That is, there exists a policy, π^* such that, when $a_t = \pi^*(b_t)$, the expected cumulative reward or "value function" is maximized for the POMDP [14, 15]. Given the POMDP model, each subsequent belief can be calculated using Bayes' rule according to

$$b'(s') = \frac{\int_{s\in\mathcal{S}} \mathcal{Z}(o \mid s, a, s')\mathcal{T}(s' \mid s, a) b(s) ds}{\int_{s'\in\mathcal{S}} \int_{s\in\mathcal{S}} \mathcal{Z}(o \mid s, a, s')\mathcal{T}(s' \mid s, a) b(s) ds ds'}. \tag{1.2}$$

When the state space is discrete, the integrals may be replaced with sums.

Generative Models

or many problems, it can be difficult to explicitly determine or represent the probability distributions $\mathcal{T}$ or $\mathcal{Z}$. Some solution approaches, however, only require samples from the state transitions and observations. A generative model, G, stochastically generates a new state, reward, and observation in the partially observable case, given the current state and action, that is $s', r = G(s, a)$ for an MDP, or $s', o, r = G(s, a)$ for a POMDP. A generative model implicitly defines $\mathcal{T}$ and $\mathcal{Z}$, even when they cannot be explicitly represented.

Belief MDP

Every POMDP is equivalent to an MDP where the state space of the MDP is the space of possible beliefs. The reward function of this "belief MDP" is the expectation of the state-action reward function with respect to the belief. The Bayesian update of the belief serves as a generative model for the belief space MDP.

1.2.4 Value Iteration

1.2.5 Monte Carlo Tree Search

MCTS is an effective and widely studied algorithms for online decision-making [6]. It works by incrementally creating a policy tree consisting of alternating layers of state nodes and action nodes and estimating the state-action value function, $Q(s, a)$, at each of the action nodes. Only a generative model, G, is required by the algorithm. The tree is constructed by running n Monte Carlo simulations with four phases, although there are many variations of this algorithm.

1. *Search.* In the initial phase of the simulation, the policy defined by the tree is used. At each state node, a selection criterion based on Q is used to choose a favorable action, and the tree is traversed through the node and to the next state node determined by G.

2. *Expansion.* Eventually, the simulation reaches an action node that does not have

any children. At this point, a new state is sampled with G and a corresponding node created along with children corresponding to each action.

3. *Rollout.* After the expansion step, the simulation is continued with a rollout policy, often consisting of randomly selected actions, until the future accumulated reward will be negligible because of the compounding discount factor.

4. *Update.* Once the simulation has terminated, the estimates of $Q(s, a)$ at each of the visited action nodes are updated with the discounted reward received during the simulation after visiting the node.

This approach builds a tree asymmetrically favoring regions of the state and action spaces that will be visited when the optimal policy is executed.

Upper Confidence Trees

The selection criterion used to choose actions in the search phase is very important. It must balance favoring actions with large Q values that are expected to yield good results with exploration of new actions. The most widely used approach for this is known as the upper confidence bound for trees (UCT) algorithm. At each state node, it chooses the action that maximizes the upper confidence bound

$$UCB(s, a) = Q(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}} \tag{1.3}$$

where $N(s, a)$ is the number of times the action node has been visited, $N(s) = \sum_{a \in \mathcal{A}} N(s, a)$, and c is a problem-specific parameter that governs the amount of exploration in the tree. The second term causes the algorithm to favor actions that have been taken less often.

Double Progressive Widening

In cases where the action and state spaces are large or continuous, the MCTS algorithm will produce trees that are very shallow. In fact, if the action space is continuous, the UCT algorithm will never try the same action twice (observe that, if

$N(s, a) = 0$ then $UCB(s, a)$ in (1.3) is infinite, so untried actions are always favored). Moreover, if the state space is continuous and the transition probability density is finite, the probability of sampling the same state twice from G is zero. Because of this, simulations will never pass through the same state node twice and a tree below the first layer of state nodes will never be constructed.

In progressive widening, the number of children of a node is artificially limited to kN^α where N is the number of times the node has been visited and k and α are parameters chosen for the problem [8]. Originally, progressive widening was applied to the action space, and was found to be especially effective when a set of preferred actions was tried first [6]. The term *double* progressive widening refers to progressive widening in both the state and action space. When the number of state nodes is greater than the limit, instead of simulating a new state transition, one of the previously generated states is chosen with probability proportional to the number of times it has been previously generated.

1.2.6 Particle Filtering

Aside from a few special cases, for example when the system dynamics are linear and the transition and observation distributions are Gaussian, the integrals in the Bayesian belief update (1.2) are impossible or difficult to solve analytically. Thus, numerical approaches must be used. A popular technique for this is particle filtering, usually incorporating domain-specific heuristic modifications to prevent problems such as particle depletion [31].

Sequential importance resampling, one of the most common and effective variations, requires a state generative model, G_s that can sample from the transition distribution and an explicit representation of the observation distribution, $\mathcal{Z}(\cdot \mid s, a, s')$, which is often easier to represent than the transition distribution. The belief is approximated with a set of m particles, $\{s_i\}_{i=1}^m$ and associated weights, $\{w_i\}_{i=1}^m$. The probability of each state is approximated by the sum of the weights corresponding to particles with that state value,

$$b(s) \approx \sum_{i=1}^{m} w_i \delta_s(s_i) \tag{1.4}$$

where $\delta_s(\cdot)$ is the Dirac delta function centered at s. A belief update is approximated by sampling m states $\{\tilde{s}_i\}_{i=1}^m$ from the collection of particles with probability proportional to the associated weight, generating a particle, $s'_i = G(\tilde{s}_i, a)$ for each of these states, and finally setting the new weight proportional to the probability of generating the measured observation with this state, $w'_i \propto \mathcal{Z}(o \mid \tilde{s}_i, a, s'_i)$.

1.2.7 Approximate Solutions to POMDPs

Considerable progress has been made in solving large POMDPs. Initially, exact offline solutions to problems with only a few discrete states, actions, and observations were sought by using value iteration and taking advantage of the convexity of the value function [14], although solutions to larger problems were also explored using Monte Carlo simulation and interpolation between belief states [30]. Many effective offline planners for discrete problems use point based value iteration, where a selection of points in the belief space are used for value function approximation, [16]. Offline solutions for problems with continuous state and observation spaces have also been proposed [4, 5].

There are also various solution approaches that are applicable to specific classes of POMDPs, including continuous problems. For example, Platt et al. [23] simplify planning in large domains by assuming that the most likely observation will always be received, which can provide an acceptable approximation in some problems with unimodal observation distributions. Morere, Marchant, and Ramos [21] solve a monitoring problem with continuous spaces with a Gaussian process belief update. Hoey and Poupart [12] propose a method for partitioning large observation spaces without information loss, but demonstrate the method only on small state and action spaces that have a modest number of conditional plans. Other methods involve motion-planning techniques [20, 24, 7]. In particular, Agha-Mohammadi, Chakravorty, and

Amato [1] present a method to take advantage of the existence of a stabilizing controller in belief space planning. Van Den Berg, Patil, and Alterovitz [32] perform local optimization with respect to uncertainty on a pre-computed path, and Indelman, Carlone, and Dellaert [13] devise a hierarchical approach that handles uncertainty in both the robot's state and the surrounding environment.

General purpose online algorithms for POMDPs have also been proposed. Many early online algorithms focused on point-based belief tree search with heuristics for expanding the trees [25]. The introduction of POMCP [27] caused a pivot toward the simple and fast technique of using the same simulations for decision-making and using beliefs implicitly represented as unweighted collections of particles. Determinized sparse partially observable tree (DESPOT) is a similar approach that attempts to achieve better performance by analyzing only a small number of random outcomes in the tree [28]. Adaptive belief tree (ABT) was designed specifically to accommodate changes in the environment without having to replan from scratch [17].

Chapter 2

Trusted and Optimized Collision Avoidance for Unmanned Aerial Vehicles

2.1 Collision Avoidance for Unmanned Aerial Vehicles

2.2 MDP Model

2.3 Trusted Resolution Logic

2.4 Approximate Value Iteration

2.4.1 Post-decision states

2.4.2 Algorithm

2.4.3 Features

2.5 Results

Chapter 3

The Value of Planning with the Internal State of Traffic Participants in Autonomous Freeway Driving

3.1 Human-Robot Interaction in Autonomous Driving

3.2 Freeway Driving POMDP

3.2.1 IDM-Mobil Driver Model

3.2.2 Parameter Correlation

3.2.3 Reward Function

3.3 Solution Approaches

3.4 Results

Chapter 4

Online Algorithms for POMDPs with Continuous Observation Spaces

4.1 Background

Although the research surveyed in Section 1.2.7 has yielded effective solution techniques for many classes of POMDPs, there remains a need for simple, general purpose online solvers that can handle continuous spaces, especially continuous observation spaces. This chapter addresses that need.

Many previous methods can easily handle continuous state spaces [11], but they must be modified to extend to domains with continuous action or observation spaces. Though DESPOT has demonstrated effectiveness on some large problems, since it uses unweighted particle beliefs in its search tree, it struggles with continuous information gathering problems as will be shown in Section 4.5. ABT has been extended to use generalized pattern search for selecting locally optimal continuous actions, an approach which is especially effective in problems where high precision is important [26], but also uses unweighted particle beliefs. Continuous observation Monte Carlo tree search (COMCTS) constructs observation classification trees to automatically partition the observation space in a POMCP-like approach, however it did not perform much better than a Monte Carlo rollout approach in experiments [22].

The algorithms discussed in this chapter are derivatives of the partially observable Monte Carlo planning algorithm (POMCP), which is based on MCTS (Section 1.2.5). A conceptually straightforward way to solve a POMDP using MCTS is to apply it to the corresponding belief MDP. Indeed, many tree search techniques have been applied to POMDP problems in this way [25]. However, when the Bayesian belief update is used, this approach is computationally expensive. POMCP and its successors, DESPOT and ABT, can tackle problems many times larger than their predecessors because they use state trajectory simulations, rather than full belief trajectories, to build the tree.

Each of the nodes in a POMCP tree corresponds to a history proceeding from the root belief and terminating with an action or observation. In the search phase of POMCP tree construction, state trajectories are simulated through this tree. At each action node, the rewards from the simulations that pass through the node are used to estimate the Q function. This simple approach has been shown to work well for large discrete problems [27]. However, when the action or observation space is continuous, the tree degenerates and does not extend beyond a single layer of nodes because each new simulation produces a new branch.

4.2 Structure and Notation

The three algorithms in this section share a common structure. For all algorithms, the entry point for the decision making process is the Plan procedure, which takes the current belief, b, as an input (Plan differs slightly for PFT-DPW in Algorithm 2). The algorithms also share the same ActionProgWiden function to control progressive widening of the action space. These components are listed in Listing 1. The difference between the algorithms is in the Simulate function.

The following variables are used in the listings and text: h represents a history $(b, a_1, o_1, \ldots a_k, o_k)$, and ha and hao are shorthand for histories with a and (a, o) appended to the end, respectively; d is the depth to explore, with $d_{\max}$ the maximum depth; C is a list of the children of a node (along with the reward in the case of

Listing 1 Common procedures

```
procedure PLAN($b$)
    for $i \in 1 : n$ do
        $s \leftarrow$ sample from $b$
        SIMULATE($s, b, d_{\max}$)
    return $\arg\max_{a} Q(ba)$
procedure ACTIONPROGWIDEN($h$)
    if $|C(h)| \leq k_a N(h)^{\alpha_a}$ then
        $a \leftarrow$ NEXTACTION($h$)
        $C(h) \leftarrow C(h) \cup \{a\}$
    return $\arg\max_{a \in C(h)} Q(ha) + c\sqrt{\frac{\log N(h)}{N(ha)}}$
```

PFT-DPW); N is a count of the number of visits; and M is a count of the number of times that a history has been generated by the model. The list of states associated with a node is denoted B, and W is a list of weights corresponding to those states. Finally, $Q(ha)$ is an estimate of the value of taking action a after observing history h. C, N, M, B, W, and Q are all implicitly initialized to 0 or $\emptyset$. The ROLLOUT procedure, runs a simulation with a default rollout policy, which can be based on the history or fully observed state for d steps and returns the discounted reward.

4.3 POMCP-DPW

The first algorithm that we consider is POMCP with double progressive widening (POMCP-DPW). In this algorithm, listed in Algorithm 1, the number of new children sampled from any node in the tree is limited by DPW using the parameters k_a, α_a, k_o, and α_o. In the case where the simulated observation is rejected (line 14), the tree search is continued with an observation selected in proportion to the number of times, M, it has been previously simulated (line 15) and a state is sampled from the associated belief (line 16).

This algorithm obtained remarkably good solutions for a very large autonomous

Algorithm 1 POMCP-DPW

```
procedure SIMULATE($s$, $h$, $d$)
    if $d = 0$ then
        return $0$
    $a \leftarrow$ ACTIONPROGWIDEN($h$)
    if $|C(ha)| \leq k_o N(ha)^{\alpha_o}$ then
        $s', o, r \leftarrow G(s, a)$
        $C(ha) \leftarrow C(ha) \cup \{o\}$
        $M(hao) \leftarrow M(hao) + 1$
        append $s'$ to $B(hao)$
        if $M(hao) = 1$ then
            $total \leftarrow r + \gamma$ROLLOUT$(s', hao, d-1)$
        else
            $total \leftarrow r + \gamma$SIMULATE$(s', hao, d-1)$
    else
        $o \leftarrow$ select $o \in C(ha)$ w.p. $\frac{M(hao)}{\sum_o M(hao)}$
        $s' \leftarrow$ select $s' \in B(hao)$ w.p. $\frac{1}{|B(hao)|}$
        $r \leftarrow R(s, a, s')$
        $total \leftarrow r + \gamma$SIMULATE$(s', hao, d-1)$
    $N(h) \leftarrow N(h) + 1$
    $N(ha) \leftarrow N(ha) + 1$
    $Q(ha) \leftarrow Q(ha) + \frac{total - Q(ha)}{N(ha)}$
    return $total$
```

freeway driving POMDP with multiple vehicles (up to 40 continuous fully observable state dimensions and 72 continuous correlated partially observable state dimensions) [29]. To our knowledge, that is the first work applying progressive widening to POMCP, and it does not contain a detailed description of the algorithm or any theoretical or experimental analysis other than the driving application.

This algorithm may converge to the optimal solution for POMDPs with discrete observation spaces; however, on continuous observation spaces, POMCP-DPW is suboptimal. In particular, it finds a QMDP policy, that is, the solution under the assumption that the problem becomes fully observable after one time step [18, 15]. In fact, for a modified version of POMCP-DPW, it is easy to prove analytically that it will converge to such a policy. This is expressed formally in Theorem 1 below. A complete description of the modified algorithm and problem requirements including the definitions of polynomial exploration, the regularity hypothesis for the problem, and exponentially sure convergence are given in Appendix A.

Definition 1 (QMDP value)**.** *Let $Q_{MDP}(s,a)$ be the optimal state-action value function assuming full observability starting by taking action a in state s. The* QMDP value *at belief b, $Q_{MDP}(b,a)$, is the expected value of $Q_{MDP}(s,a)$ when s is distributed according to b.*

Theorem 1 (Modified POMCP-DPW convergence to QMDP)**.** *If a bounded-horizon POMDP meets the following conditions: 1) the state and observation spaces are continuous with a finite observation probability density function, and 2) the regularity hypothesis is met, then modified POMCP-DPW will produce a value function estimate, $\hat{Q}$, that converges to the QMDP value for the problem. Specifically, there exists a constant $C > 0$, such that after n iterations,*

$$\left|\hat{Q}(b,a) - Q_{MDP}(b,a)\right| \leq \frac{C}{n^{1/(10d_{\max}-7)}}$$

exponentially surely in n, for every action a.

A proof of this theorem that leverages work by Auger, Couetoux, and Teytaud [3] is given in Appendix A, but we provide a brief justification here. The key is that belief nodes will contain only a single state particle (see Fig. 4.1). This is because, since the observation space is continuous with a finite density function, the generative model will (with probability one) produce a unique observation o each time it is queried. Thus, for every generated history h, only one state will ever be inserted into $B(h)$ (line 9, Algorithm 1), and therefore h is merely an alias for that state. Since each belief node corresponds to a state, the solver is actually solving the fully observable MDP at every node except the root node, leading to a QMDP solution.

As a result of Theorem 1, the action chosen by modified POMCP-DPW will match a QMDP policy (a policy of actions that maximize the QMDP value) with high precision exponentially surely (see Corollary 1 of Auger, Couetoux, and Teytaud [3]). For many problems this is a very useful solution,[1] but since it neglects the value of information, a QMDP policy is suboptimal for problems where information gathering is important [18, 15].

Although Theorem 1 is only theoretically applicable to the modified version of POMCP-DPW, it helps explain the behavior of other solvers. Modified POMCP-DPW, POMCP-DPW, DESPOT, and ABT all share the characteristic that a belief node can only contain two states if they generated exactly the same observation. Since this is an event with zero probability for a continuous observation space, these solvers exhibit suboptimal, often QMDP-like, behavior. The experiments in Section 4.5 show this for POMCP-DPW and DESPOT, and this is presumably the case for ABT as well.

4.4 PFT-DPW

Another algorithm that one might consider for solving continuous POMDPs online is MCTS-DPW on the equivalent belief MDP. Since the Bayesian belief update is usually computationally intractable, a particle filter is used. This new approach will be referred to as particle filter trees with double progressive widening (PFT-DPW). It is shown in Algorithm 2, where $G_{\text{PF}(m)}(b, a)$ is a particle filter belief update performed

[1]Indeed, a useful online QMDP tree search algorithm could be created by deliberately constructing a tree with a single root belief node and fully observable state nodes below it.

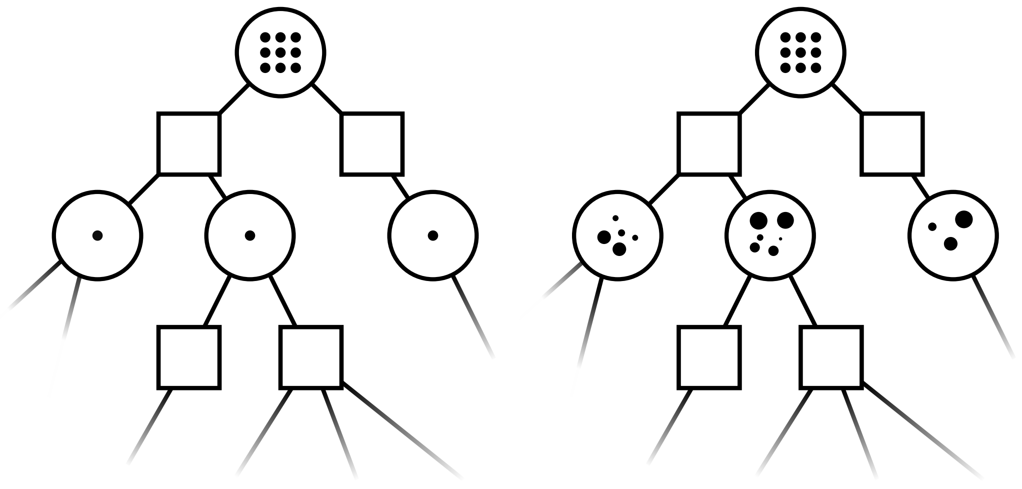

(a) POMCP-DPW Tree (b) POMCPOW Tree

Figure 4.1: Tree Structure Comparison. Each square is an action node, and each unfilled circle is an observation node. Each black dot corresponds to a state particle with the size representing its weight. In continuous observation spaces, the beliefs in a POMCP-DPW tree degenerate to a single particle, while POMCPOW maintains weighted particle mixture beliefs.

with a simulated observation and m state particles which approximates the belief MDP generative model. The authors are not aware of any mention of this algorithm in prior literature, but it is very likely that MCTS with particle filters has been used before without double progressive widening under another name.

PFT-DPW is fundamentally different from POMCP and POMCPOW because it relies on simulating approximate belief trajectories instead of state trajectories. This distinction also allows it to be applied to problems where the reward is a function of the belief rather than the state such as pure information-gathering problems [9, 2].

The primary shortcoming of this algorithm is that the number of particles in the filter, m, must be chosen a-priori and is static throughout the tree. Each time a new belief node is created, an $\mathcal{O}(m)$ particle filter update is performed. If m is too small, the beliefs may miss important states, but if m is too large, constructing the tree is expensive. Fortunately, the experiments in Section 4.5 show that it is often easy to choose m in practice; for all the problems we studied, a value of $m = 20$ resulted in good performance.

Algorithm 2 PFT-DPW

```
1: procedure PLAN($b$)
2:     for $i \in 1 : n$ do
3:         SIMULATE($b, d_{\max}$)
4:     return $\arg\max_a Q(ba)$
5: procedure SIMULATE($b$, $d$)
6:     if $d = 0$ then
7:         return 0
8:     $a \leftarrow$ ACTIONPROGWIDEN($b$)
9:     if $|C(ba)| \le k_o N(ba)^{\alpha_o}$ then
10:        $b', r \leftarrow G_{\text{PF}(m)}(b, a)$
11:        $C(ba) \leftarrow C(ba) \cup \{(b', r)\}$
12:        $total \leftarrow r + \gamma$ROLLOUT($b', d-1$)
13:    else
14:        $b', r \leftarrow$ sample uniformly from $C(ba)$
15:        $total \leftarrow r + \gamma$SIMULATE($b', d-1$)
16:    $N(b) \leftarrow N(b) + 1$
17:    $N(ba) \leftarrow N(ba) + 1$
18:    $Q(ba) \leftarrow Q(ba) + \frac{total - Q(ba)}{N(ba)}$
19:    return $total$
```

Algorithm 3 POMCPOW

```
 1: procedure SIMULATE($s$, $h$, $d$)
 2:     if $d = 0$ then
 3:         return 0
 4:     $a \leftarrow$ ACTIONPROGWIDEN($h$)
 5:     $s', o, r \leftarrow G(s, a)$
 6:     if $|C(ha)| \leq k_o N(ha)^{\alpha_o}$ then
 7:         $M(hao) \leftarrow M(hao) + 1$
 8:     else
 9:         $o \leftarrow$ select $o \in C(ha)$ w.p. $\frac{M(hao)}{\sum_o M(hao)}$
10:     append $s'$ to $B(hao)$
11:     append $\mathcal{Z}(o \mid s, a, s')$ to $W(hao)$
12:     if $o \notin C(ha)$ then                                    ▷ new node
13:         $C(ha) \leftarrow C(ha) \cup \{o\}$
14:         $total \leftarrow r + \gamma$ROLLOUT$(s', hao, d - 1)$
15:     else
16:         $s' \leftarrow$ select $B(hao)[i]$ w.p. $\frac{W(hao)[i]}{\sum_{j=1}^{m} W(hao)[j]}$
17:         $r \leftarrow R(s, a, s')$
18:         $total \leftarrow r + \gamma$SIMULATE$(s', hao, d - 1)$
19:     $N(h) \leftarrow N(h) + 1$
20:     $N(ha) \leftarrow N(ha) + 1$
21:     $Q(ha) \leftarrow Q(ha) + \frac{total - Q(ha)}{N(ha)}$
22:     return $total$
```

4.4.1 POMCPOW

In order to address the suboptimality of POMCP-DPW, we now propose a new algorithm, POMCPOW, shown in Algorithm 3. In this algorithm, the belief updates are weighted, but they also expand gradually as more simulations are added. Furthermore, since the richness of the belief representation is related to the number of times the node is visited, beliefs that are more likely to be reached by the optimal policy have more particles. At each step, the simulated state is inserted into the weighted particle collection that represents the belief (line 10), and a new state is sampled from that belief (line 16). A simple illustration of the tree is shown in Figure 4.1 to contrast with a POMCP-DPW tree. Because the resampling in line 16 can be efficiently implemented with binary search, the computational complexity is $\mathcal{O}(nd\log(n))$.

4.4.2 Discretization

Discretization is perhaps the most straightforward way to deal with continuous observation spaces. The results in Table 4.1 show that this approach is only sometimes effective. Figure 4.2 shows the performance at different discretization granularities for the Light Dark and Sub Hunt problems.

Since the Light Dark domain has only a single observation dimension, it is easy to discretize. In fact, POMCP with fine discretization outperforms POMCPOW. However, discretization is only effective at certain granularities, and this is highly dependent on the solver and possibly hyperparameters. In the Sub Hunt problem, with its high-dimensional observation, discretization is not effective at any granularity. In Van Der Pol tag, both the action and observation spaces must be discretized. Due to the high dimensionality of the observation space, similar to Sub Hunt, no discretization that resulted in good performance was found.

4.4.3 Observation Distribution Requirement

It is important to note that, while POMCP, POMCP-DPW, and DESPOT only require a generative model of the problem, both POMCPOW and PFT-DPW require a way to query the relative likelihood of different observations ($\mathcal{Z}$ in line 11). One may object that this will limit the application of POMCPOW to a small class of POMDPs, but we think it will be an effective tool in practice for two reasons.

First, this requirement is no more stringent than the requirement for a standard importance resampling particle filter, and such filters are used widely, at least in the field of robotics that the authors are most familiar with. Moreover, if the observation model is complex, an approximate model may be sufficient.

Second, given the implications of Theorem 1, it is difficult to imagine a tree-based decision-making algorithm or a robust belief updater that does not require some way of measuring whether a state belongs to a belief or history. The observation model is a straightforward and standard way of specifying such a measure. Finally, in practice, except for the simplest of problems, using POMCP or DESPOT to repeatedly observe and act in an environment already requires more than just a generative model.

Table 4.1: Experimental Results

	Multilane (C, D, C)
POMCPOW	0.3 ± 0.0 🟩
PFT-DPW	
QMDP	0.3 ± 0.0 🟩
POMCP-DPW	
DESPOT	0.4 ± 0.0 🟩
POMCP$^{\mathrm{D}}$	
DESPOT$^{\mathrm{D}}$	

The three C or D characters after the solver indicate whether the state, action, and observation spaces are continuous or discrete, respectively. For continuous problems, solvers with a superscript D were run on a version of the problem with discretized action and observation spaces, but they interacted with continuous simulations of the problem.

For example, the authors of the original paper describing POMCP [27] use heuristic particle reinvigoration in lieu of an observation model and importance sampling.

4.5 Experiments

Numerical simulation experiments were conducted to evaluate the performance of POMCPOW and PFT-DPW compared to other solvers. The open source code for the experiments is built on the POMDPs.jl framework [10] and is hosted at `https://github.com/zsunberg/ContinuousPOMDPTreeSearchExperiments.jl`. In all experiments, the solvers were limited to 1 second of computation time per step. Belief updates were accomplished with a particle filter independent of the planner, and no part of the tree was saved for re-use on subsequent steps. Hyperparameter values are shown in Section 4.5.5.

4.5.1 Laser Tag

The Laser Tag benchmark is taken directly from the work of Somani et al. [28] and included for the sake of calibration. DESPOT outperforms the other methods. The score for DESPOT differs slightly from that reported by Somani et al. [28] likely because of bounds implementation differences. POMCP performs much better than reported by Somani et al. [28] because this implementation uses a state-based rollout policy.

4.5.2 Light Dark

In the Light Dark domain, the state is an integer, and the agent can choose how to move deterministically ($s' = s + a$) from the action space $\mathcal{A} = \{-10, -1, 0, 1, -10\}$. The goal is to reach the origin. If action 0 is taken at the origin, a reward of 100 is given and the problem terminates; If action 0 is taken at another location, a penalty of -100 is given. There is a cost of -1 at each step before termination. The agent receives a more accurate observation in the "light" region around $s = 10$. Specifically, observations are continuous ($\mathcal{O} = \mathbb{R}$) and normally distributed with standard deviation $\sigma = |s - 10|$.

Table 4.1 shows the mean reward from 1000 simulations for each solver, and Fig. 4.3 shows an example experiment. The optimal strategy involves moving toward the light region and localizing before proceeding to the origin. QMDP and solvers predicted to behave like QMDP attempt to move directly to the origin, while POMCPOW and PFT-DPW perform better. In this one-dimensional case, discretization allows POMCP to outperform all other methods and DESPOT to perform well, but in subsequent problems where the observation space has more dimensions, discretization does not provide the same performance improvement (see Section 4.4.2).

4.5.3 Sub Hunt

In the Sub Hunt domain, the agent is a submarine attempting to track and destroy an enemy sub. The state and action spaces are discrete so that QMDP can be used to solve the problem for comparison. The agent and the target each occupy a cell of a 20 by 20 grid. The target is either aware or unaware of the agent and seeks to reach a particular edge of the grid unknown to the agent ($\mathcal{S} = \{1, .., 20\}^4 \times \{\text{aware}, \text{unaware}\} \times \{N, S, E, W\}$). The target stochastically moves either two steps towards the goal or one step forward and one to the side. The agent has six actions, move three steps north, south, east, or west, engage the other submarine, or ping with active sonar.

If the agent chooses to engage and the target is unaware and within a range of 2, a hit with reward 100 is scored; The problem ends when a hit is scored or the target reaches its goal edge.

An observation consists of 8 sonar returns ($\mathcal{O} = \mathbb{R}^8$) at equally-spaced angles that give a normally distributed estimate ($\sigma = 0.5$) of the range to the target if the target is within that beam and a measurement with higher variance if it is not. The range of the sensors depends on whether the agent decides to use active sonar. If the agent does not use active sonar it can only detect the other submarine within a radius of 3, but pinging with active sonar will detect at any range. However, active sonar alerts the target to the presence of the agent, and when the target is aware, the hit probability when engaging drops to 60%.

Table 4.1 shows the mean reward for 1000 simulations for each solver. The optimal strategy includes using the active sonar, but previous approaches have difficulty determining this because of the reduced engagement success rate. The PFT-DPW approach has the best score, followed closely by POMCPOW. All other solvers have similar performance to QMDP.

4.5.4 Van Der Pol Tag

The final experimental problem is called Van Der Pol tag and has continuous state, action, and observation spaces. In this problem an agent moves through 2D space to try to tag a target ($\mathcal{S} = \mathbb{R}^4$) that has a random unknown initial position in $[-4, 4] \times [-4, 4]$. The agent always travels at the same speed, but chooses a direction of travel and whether to take an accurate observation ($\mathcal{A} = [0, 2\pi) \times \{0, 1\}$). The observation again consists of 8 beams ($\mathcal{O} = \mathbb{R}^8$) that give measurements to the target. Normally, these measurements are too noisy to be useful ($\sigma = 5$), but, if the agent chooses an accurate measurement with a cost of 5, the observation has low noise ($\sigma = 0.1$). The agent is blocked if it comes into contact with one of the barriers that stretch from 0.2 to 3.0 in each of the cardinal directions (see Fig. 4.4), while the target can move freely through. There is a cost of 1 for each step, and a reward of 100 for tagging the target (being within a distance of 0.1).

The target moves following a two dimensional form of the Van Der Pol oscillation defined by the differential equations

$$\dot{x} = \mu \left(x - \frac{x^3}{3} - y \right) \quad \text{and} \quad \dot{y} = \frac{1}{\mu} x,$$

where $\mu = 2$. Gaussian noise ($\sigma = 0.05$) is added to the position at the end of each step. Runge-Kutta fourth order integration is used to propagate the state.

This problem has several challenging features that might be faced in real-world applications. First, the state transitions are more computationally expensive because of the numerical integration. Second, the continuous state space and obstacles make it difficult to construct a good heuristic rollout policy, so random rollouts are used. Table 4.1 shows the mean reward for 1000 simulations of this problem for each solver. Since a POMCPOW iteration requires less computation than a PFT-DPW iteration, POMCPOW simulates more random rollouts and thus performs slightly better.

4.5.5 Hyperparameters

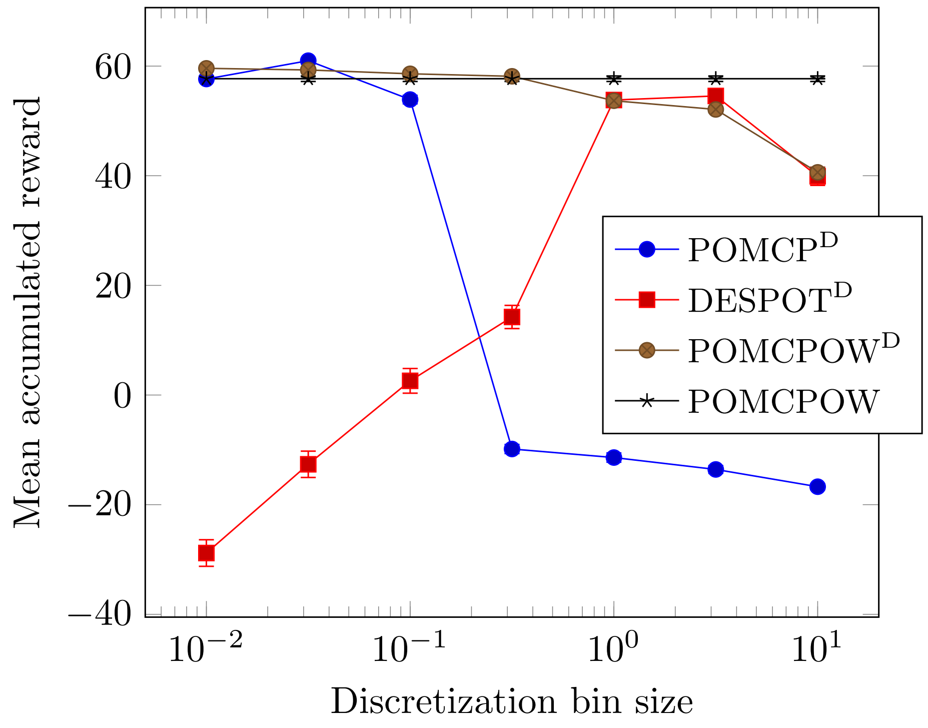
Hyperparameters for POMCPOW and PFT-DPW were chosen using the cross entropy method [19], but exact tuning was not a high priority and some parameters were re-used across solvers so the parameters may not be perfectly optimized. The values used in the experiments are shown in Table 4.2. There are not enough experiments to draw broad conclusions about the hyperparameters, but it appears that performance is most sensitive to the exploration constant, c.

The values for the observation widening parameters, k_o and α_o, were similar for all the problems in this work. A small α_o essentially limits the number of observations to a static number k_o, resulting in behavior reminiscent of sparse UCT [6], preventing unnecessary widening and allowing the tree to grow deep. This seems to work well in practice with the branching factor (k_o) set to values between 2 and 8, and suggests that it may be sufficient to limit the number of children to a fixed number rather than do progressive widening in a real implementation.

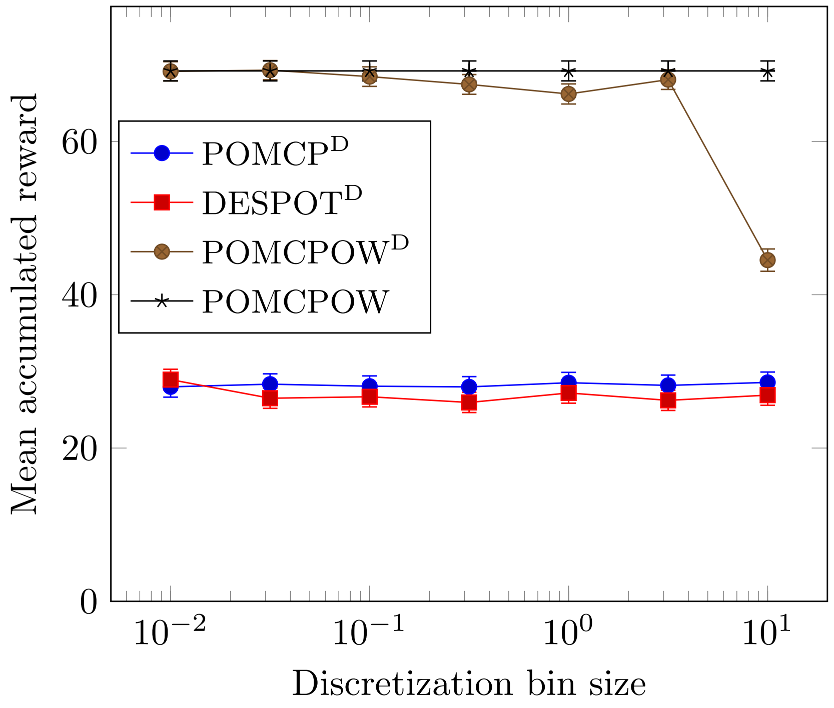
Table 4.2: Hyperparameters used in experiments

	Laser Tag	Light Dark	Sub Hunt	VDP Tag
POMCPOW				
c	26.0	90.0	17.0	110.0
k_a	–	–	–	30.0
α_a	–	–	–	1/30
k_o	4.0	5.0	6.0	5.0
α_o	1/35	1/15	1/100	1/100
PFT-DPW				
m	20	20	20	20
c	26.0	100.0	100.0	70.0
k_a	–	–	–	20.0
α_a	–	–	–	1/25
k_o	4.0	4.0	2.0	8.0
α_o	1/35	1/10	1/10	1/85

For problems with discrete actions, all actions are considered and k_a and α_a are not needed.



(a) Light Dark



(b) Sub Hunt

Figure 4.2: Discretization granularity studies

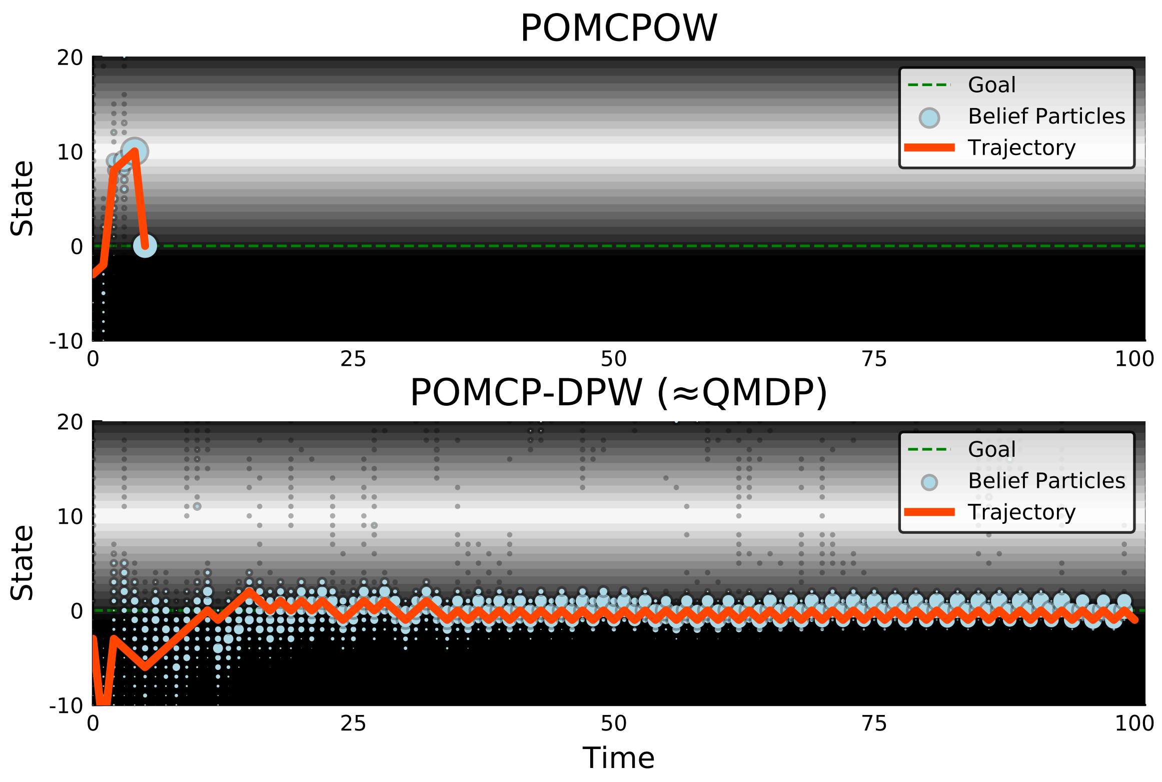


Figure 4.3: Example trajectories in the Light Dark domain. POMCPOW travels to the light region and accurately localizes before moving to the goal. POMCP-DPW displays QMDP-like behavior: it is unable to localize well enough to take action 0 with confidence. The belief particles far away from 0 in the POMCP-DPW plot are due to particle reinvigoration that makes the filter more robust.

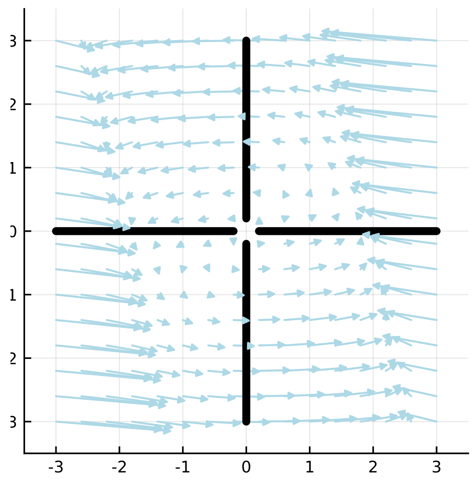


Figure 4.4: Van Der Pol tag problem. The arrows show the target differential equation, and the thick black lines represent the barriers.

Appendix A

Proof of Theorem 1

A version of Monte Carlo tree search with double progressive widening has been proven to converge to the optimal value function on fully observable MDPs by Auger, Couetoux, and Teytaud [3]. We utilize this proof to show that POMCP-DPW converges to a solution that is sometimes suboptimal.

First we establish some preliminary definitions taken directly from Auger, Couetoux, and Teytaud [3].

Definition 2 (Regularity Hypothesis)**.** *The* Regularity hypothesis *is the assumption that for any $\Delta > 0$, there is a non zero probability to sample an action that is optimal with precision Δ. More precisely, there is a $\theta > 0$ and a $p > 1$ (which remain the same during the whole simulation) such that for all $\Delta > 0$,*

$$\begin{aligned} &Q(ha) \geq Q^*(h) - \Delta \\ &\qquad \textit{with probability at least } \min(1, \theta\Delta^p). \end{aligned} \tag{A.1}$$

Definition 3 (Exponentially sure in n)**.** *We say that some property depending on an integer n is exponentially sure in n if there exists positive constants C, h, and η such that the probability that the property holds is at least*

$$1 - C\exp(-hn^\eta).$$

In order for the proof from Auger, Couetoux, and Teytaud [3] to apply, the following four minor modifications to the POMCP-DPW algorithm must be made:

1. Instead of the usual logarithmic exploration, use *polynomial exploration*, that is, select actions based on the criterion

   $$Q(ha) + \sqrt{\frac{N(h)^{e_d}}{N(ha)}} \tag{A.2}$$

   as opposed to the traditional criterion

   $$Q(ha) + c\sqrt{\frac{\log N(h)}{N(ha)}}, \tag{A.3}$$

   and create a new node for progressive widening when $\lfloor N^\alpha \rfloor > \lfloor (N-1)^\alpha \rfloor$ rather than when the number of children exceeds kN^α.

2. Instead of performing rollout simulations, keep creating new single-child nodes until the maximum depth is reached.

3. In line 15, instead of selecting an observation randomly, select the observation that has been visited least proportionally to how many times it has been visited.

4. Use the depth-dependent coefficient values in Table 1 from Auger, Couetoux, and Teytaud [3] instead of choosing static values.

This version of the algorithm will be referred to as "modified POMCP-DPW". The algorithm with these changes is listed in Algorithm 4.

We now define the "QMDP value" that POMCP-DPW converges to (this is repeated from the main text of the paper) and prove a preliminary lemma.

Definition 4 (QMDP value)**.** *Let $Q_{MDP}(s,a)$ be the optimal state-action value function assuming full observability starting by taking action a in state s. The* QMDP value *at belief b, $Q_{MDP}(b,a)$, is the expected value of $Q_{MDP}(s,a)$ when s is distributed according to b.*

Lemma 1. *If POMCP-DPW or modified POMCP-DPW is applied to a POMDP with a continuous observation space and observation probability density functions that are finite everywhere, then each history node in the tree will have only one corresponding state, that is $|B(h)| = 1, M(h) = 1\,\forall h$.*

Proof. Since the observation probability density function is finite, each call to the generative model will produce a unique observation with probability 1. Because of this, lines 18 and 19 of Algorithm 4 will only be executed once for each observation. □

We are now ready to restate and prove the theorem from the text.

Theorem 1 (Modified POMCP-DPW convergence to QMDP)**.** *If a bounded-horizon POMDP meets the following conditions: 1) the state and observation spaces are continuous with a finite observation probability density function, and 2) the regularity hypothesis is met, then modified POMCP-DPW will produce a value function estimate, $\hat{Q}$, that converges to the QMDP value for the problem. Specifically, there exists a constant $C > 0$, such that after n iterations,*

$$\left|\hat{Q}(b,a) - Q_{MDP}(b,a)\right| \leq \frac{C}{n^{1/(10d_{\max}-7)}}$$

exponentially surely in n, for every action a.

Proof. We prove that modified POMCP-DPW functions exactly as the Polynomial UCT (PUCT) algorithm defined by Auger, Couetoux, and Teytaud [3] applied to an augmented fully observable MDP, and hence converges to the QMDP value. We will show this by proposing incremental changes to Algorithm 4 that do not change its function that will result in an algorithm identical to PUCT.

Before listing the changes, we define the "augmented fully observable MDP" as follows: For a POMDP $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathcal{Z}, \gamma)$, and belief b, the *augmented fully observable MDP*, $\mathcal{M}$, is the MDP defined by $(\mathcal{S}_A, \mathcal{A}, \mathcal{T}_A, \mathcal{R}, \gamma)$, where

$$\mathcal{S}_A = \mathcal{S} \cup \{b\} \tag{A.4}$$

and, for all $x, x' \in \mathcal{S}_A$,

$$\mathcal{T}_A(x'|x,a) = \begin{cases} \mathcal{T}(x'|x,a) & \text{if } x \in \mathcal{S} \\ \int_S b(s)\mathcal{T}(x'|s,a)ds & \text{if } x = b \end{cases} \tag{A.5}$$

This is simply the fully observable MDP augmented with a special state representing the current belief. It is clear that the value function for this problem $Q_{\mathcal{M}}(b,a)$ is the same as the QMDP value for the POMDP, $Q_{\text{MDP}}(b,a)$. Thus, by showing that modified POMCP-DPW behaves exactly as PUCT applied to $\mathcal{M}$, we show that it estimates the QMDP values.

Consider the following modifications to Algorithm 4 that do not change its behavior when the observation space is continuous:

1. Eliminate the state count M. *Justification*: By Lemma 1, its value will be 1 for every node.

2. Remove B and replace with a mapping H from each node to a state of $\mathcal{M}$; define $H(b) = b$. *Justification*: By Lemma 1 B always contains only a single state, so H contains the same information.

3. Generate states and rewards with $G_{\mathcal{M}}$, the generative model of $\mathcal{M}$, instead of G. *Justification*: Since the state transition model for the fully observable MDP is the same as the POMDP, these are equivalent for all $s \in \mathcal{S}$.

4. Remove the s argument of Simulate. *Justification*: The sampling in line 3 is done implicitly in $G_{\mathcal{M}}$ if $h = b$, and s is redundant in other cases because h can be mapped to s through H.

The result of these changes is shown in Algorithm 5. It is straightforward to verify that this algorithm is equivalent to PUCT applied to $\mathcal{M}$. Each observation-terminated history, h, corresponds to a PUCT "decision node", z, and each action-terminated history, ha, corresponds to a PUCT "chance node", w. In other words, the observations have no meaning in the tree other than making up the histories, which are effectively just keys or aliases for the state nodes.

Since PUCT is guaranteed by Theorem 1 of Auger, Couetoux, and Teytaud [3] to converge to the optimal value function of $\mathcal{M}$ exponentially surely, POMCP-DPW is guaranteed to converge to the QMDP value exponentially surely, and the theorem is proven.

$\square$

Remark 1. *One may object that multiple histories may map to the same state through H, and thus the history nodes in a modified POMCP-DPW tree are not equivalent to state nodes in the PUCT tree. In fact, the PUCT algorithm does not check to see if a state has previously been generated by the model, so it may also contain multiple decision nodes z that correspond to the same state. Though this is not explicitly stated by the authors, it is clear from the algorithm description, and the proof still holds.*

Algorithm 4 Modified POMCP-DPW

```
1: procedure Plan($b$)
2:     for $i \in 1 : n$ do
3:         $s \leftarrow$ sample from $b$
4:         Simulate($s, b, d_{\max}$)
5:     return $\arg\max_a Q(ba)$
6: procedure ActionProgWiden($h$)
7:     if $\lfloor N(h)^{\alpha_{a,d}} \rfloor > \lfloor (N(h)-1)^{\alpha_{a,d}} \rfloor$ then
8:         $a \leftarrow$ NextAction($h$)
9:         $C(h) \leftarrow C(h) \cup \{a\}$
10:    return $\arg\max_{a \in C(h)} Q(ha) + \sqrt{\frac{N(h)^{e_d}}{N(ha)}}$
11: procedure Simulate($s$, $h$, $d$)
12:    if $d = 0$ then
13:        return 0
14:    $a \leftarrow$ ActionProgWiden($h$)
15:    if $\lfloor N(ha)^{\alpha_{o,d}} \rfloor > \lfloor (N(ha)-1)^{\alpha_{o,d}} \rfloor$ then
16:        $s', o, r \leftarrow G(s, a)$
17:        $C(ha) \leftarrow C(ha) \cup \{o\}$
18:        $M(hao) \leftarrow M(hao) + 1$
19:        append $s'$ to $B(hao)$
20:    else
21:        $o \leftarrow \arg\min_{o \in C(ha)} N(hao)/M(hao)$
22:        $s' \leftarrow$ select $s' \in B(hao)$ w.p. $\frac{1}{|B(hao)|}$
23:        $r \leftarrow R(s, a, s')$
24:    $total \leftarrow r + \gamma$Simulate($s', hao, d-1$)
25:    $N(h) \leftarrow N(h) + 1$
26:    $N(ha) \leftarrow N(ha) + 1$
27:    $Q(ha) \leftarrow Q(ha) + \frac{total - Q(ha)}{N(ha)}$
28:    return $total$
```

Algorithm 5 Modified POMCP-DPW on a continuous observation space

```
1: procedure PLAN($b$)
2:     for $i \in 1 : n$ do
3:         SIMULATE($(b), d_{\max}$)
4:     return $\arg\max_a Q(ha)$
5: procedure ACTIONPROGWIDEN($h$)
6:     if $\lfloor N(h)^{\alpha_{a,d}} \rfloor > \lfloor (N(h)-1)^{\alpha_{a,d}} \rfloor$ then
7:         $a \leftarrow$ NEXTACTION($h$)
8:         $C(h) \leftarrow C(h) \cup \{a\}$
9:     return $\arg\max_{a \in C(h)} Q(ha) + \sqrt{\frac{N(h)^{e_d}}{N(ha)}}$
10: procedure SIMULATE($h$, $d$)
11:     if $d = 0$ then
12:         return 0
13:     $a \leftarrow$ ACTIONPROGWIDEN($h, d$)
14:     if $\lfloor N(ha)^{\alpha_{o,d}} \rfloor > \lfloor (N(ha)-1)^{\alpha_{o,d}} \rfloor$ then
15:         $\cdot, o, \cdot \leftarrow G(H(h), a)$
16:         $H(hao), r \leftarrow G_{\mathcal{M}}(H(h), a)$
17:         $C(ha) \leftarrow C(ha) \cup \{o\}$
18:     else
19:         $o \leftarrow \arg\min_{o \in C(ha)} N(hao)$
20:         $r \leftarrow R(H(h), a, H(hao))$
21:     $total \leftarrow r + \gamma$SIMULATE($hao, d-1$)
22:     $N(h) \leftarrow N(h) + 1$
23:     $N(ha) \leftarrow N(ha) + 1$
24:     $Q(ha) \leftarrow Q(ha) + \frac{total - Q(ha)}{N(ha)}$
25:     return $total$
```

Bibliography

[1] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. "FIRM: Feedback controller-based information-state roadmap - a framework for motion planning under uncertainty". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[2] Mauricio Araya et al. "A POMDP Extension with Belief-dependent Rewards". In: *Advances in Neural Information Processing Systems (NIPS)*. 2010. URL: `http://papers.nips.cc/paper/3971-a-pomdp-extension-with-belief-dependent-rewards.pdf`.

[3] David Auger, Adrien Couetoux, and Olivier Teytaud. "Continuous upper confidence trees with polynomial exploration–consistency". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 194–209.

[4] Haoyu Bai, David Hsu, and Wee Sun Lee. "Integrated perception and planning in the continuous space: A POMDP approach". In: *International Journal of Robotics Research* 33.9 (2014), pp. 1288–1302.

[5] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. "Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation". In: *International Conference on Machine Learning (ICML)*. 2013, pp. 370–378.

[6] Cameron B. Browne et al. "A survey of Monte Carlo tree search methods". In: *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012), pp. 1–43.

[7] Adam Bry and Nicholas Roy. "Rapidly-exploring random belief trees for motion planning under uncertainty". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 723–730.

[8] A. Couëtoux et al. "Continuous Upper Confidence Trees". In: *Learning and Intelligent Optimization*. Rome, Italy, 2011.

[9] Louis Dressel and Mykel Kochenderfer. "Efficient Decision-Theoretic Target Localization". In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2017. URL: `https://www.aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15761/15090`.

[10] Maxim Egorov et al. "POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty". In: *Journal of Machine Learning Research* 18.26 (2017), pp. 1–5. URL: `http://jmlr.org/papers/v18/16-300.html`.

[11] Alex Goldhoorn et al. "Continuous real time POMCP to find-and-follow people by a humanoid service robot". In: *IEEE-RAS International Conference on Humanoid Robots*. 2014.

[12] Jesse Hoey and Pascal Poupart. "Solving POMDPs with continuous or large discrete observation spaces". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2005, pp. 1332–1338.

[13] Vadim Indelman, Luca Carlone, and Frank Dellaert. "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments". In: *International Journal of Robotics Research* 34.7 (2015), pp. 849–882.

[14] L. P. Kaelbling, M. L. Littman, and A.R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101 (1998), pp. 99–134.

[15] Mykel J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[16] Hanna Kurniawati, David Hsu, and Wee Sun Lee. "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces." In: *Robotics: Science and Systems*. Zurich, Switzerland., 2008.

[17] Hanna Kurniawati and Vinay Yadav. "An online POMDP solver for uncertainty planning in dynamic environment". In: *Robotics Research*. Springer, 2016, pp. 611–629.

[18] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. "Learning policies for partially observable environments: Scaling up". In: *International Conference on Machine Learning (ICML)*. 1995.

[19] Shie Mannor, Reuven Rubinstein, and Yohai Gat. "The Cross Entropy Method for Fast Policy Search". In: *International Conference on Machine Learning (ICML)*. 2003, pp. 512–519.

[20] Nik A Melchior and Reid Simmons. "Particle RRT for path planning with uncertainty". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2007.

[21] Philippe Morere, Roman Marchant, and Fabio Ramos. "Bayesian Optimisation for solving Continuous State-Action-Observation POMDPs". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.

[22] Andreas Pas. "Simulation Based Planning for Partially Observable Markov Decision Processes with Continuous Observation Spaces". MA thesis. Maastricht University, 2012.

[23] Robert Platt Jr. et al. "Belief space planning assuming maximum likelihood observations". In: *Robotics: Science and Systems*. 2010.

[24] Samuel Prentice and Nicholas Roy. "The belief roadmap: Efficient planning in belief space by factoring the covariance". In: *International Journal of Robotics Research* 28.11-12 (2009), pp. 1448–1465.

[25] Stéphane Ross et al. "Online planning algorithms for POMDPs". In: *Journal of Artificial Intelligence Research* 32 (2008), pp. 663–704.

[26] Konstantin M. Seiler, Hanna Kurniawati, and Surya P. N. Singh. “An online and approximate solver for POMDPs with continuous action space”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2290–2297.

[27] David Silver and Joel Veness. “Monte-Carlo Planning in Large POMDPs”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2010.

[28] Adhiraj Somani et al. “DESPOT: Online POMDP planning with regularization”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 1772–1780.

[29] Zachary N. Sunberg, Christopher J. Ho, and J. Kochenderfer Mykel. “The Value of Inferring the Internal State of Traffic Participants for Autonomous Freeway Driving”. In: *American Control Conference (ACC)*. 2017.

[30] Sebastian Thrun. “Monte Carlo POMDPs.” In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 12. 1999, pp. 1064–1070.

[31] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[32] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. “Motion planning under uncertainty using iterative local optimization in belief space”. In: *International Journal of Robotics Research* 31.11 (2012), pp. 1263–1278.