# POMCPOW: An online algorithm for POMDPs with continuous state, action, and observation spaces

**Zachary N. Sunberg*** and **Mykel J. Kochenderfer***
Aeronautics and Astronautics Dept.
Stanford University
Stanford, CA 94305

**Abstract**

Online solvers for partially observable Markov decision processes have been applied to problems with large discrete state spaces, but continuous state, action, and observation spaces remain a challenge. This paper begins by investigating double progressive widening (DPW) as a solution to this challenge. However, we prove that this modification alone is not sufficient because the belief representations in the search tree collapse to a single particle causing the algorithm to converge to a policy that is suboptimal regardless of the computation time. The main contribution of the paper is to propose a new algorithm, POMCPOW, that incorporates DPW and weighted particle filtering to overcome this deficiency and attack continuous problems. Simulation results show that these modifications allow the algorithm to be successful where previous approaches fail.

## 1 Introduction

The partially observable Markov decision process (POMDP) is a flexible mathematical framework for representing sequential decision problems (Littman, Cassandra, and Kaelbling 1995; Thrun, Burgard, and Fox 2005). Once a problem has been formalized as a POMDP, a wide range of solution techniques can be used to solve it. In a POMDP, at each step in time, an agent selects an action, and the state of the world changes stochastically based only on the current state and action. The agent seeks to maximize the expectation of the reward, which is a function of the state and action. However, the agent cannot directly observe the state, and may only make decisions based on observations that are stochastically generated by the state.

Many *offline* methods have been developed to solve small and moderately sized POMDPs (Kurniawati, Hsu, and Lee 2008). Solving larger POMDPs generally requires the use of *online* methods (Silver and Veness 2010; Somani et al. 2013; Kurniawati and Yadav 2016). One widely used online algorithm is partially observable Monte Carlo planning (POMCP) (Silver and Veness 2010), which is an extension to Monte Carlo tree search that implicitly uses an unweighted particle filter to represent beliefs in the search tree.

* {zsunberg, mykel}@stanford.edu

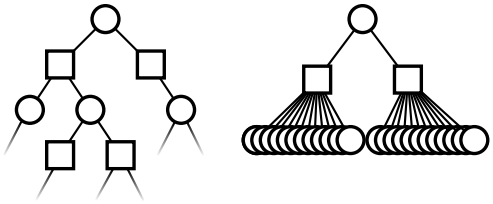


Figure 1: POMCP tree for a discrete POMDP (left), and for a POMDP with a continuous observation space (right). Because the observation space is continuous, each simulation creates a new observation node and the tree cannot extend deeper.

POMCP and other online methods can accomodate continuous state spaces, and there has been recent work on solving problems with continuous action spaces (Seiler, Kurniawati, and Singh 2015). However, there has been less progress on problems with continuous observation spaces. This paper presents a new algorithm based on POMCP, which addresses the challenge of solving POMDPs with continuous state, action, and observation spaces. We call the algorithm partially observable Monte Carlo planning with observation widening (POMCPOW).

There are two challenges that make tree search difficult in continuous spaces. The first is that, since the probability of sampling the same real number twice from a continuous random variable is zero, the width of the planning trees explodes on the first step, causing them to be too shallow to be useful (see Fig. 1). POMCPOW resolves this issue with a technique called double progressive widening (DPW) (Couëtoux et al. 2011). The second issue is that, when DPW is applied, the belief representations used by current solvers collapse to a single state particle, resulting in overconfidence. As a consequence, the solutions obtained are equivalent to QMDP policies, and there is no incentive for information gathering behavior. POMCPOW overcomes this issue by levering the observation model and using *weighted* particle mixtures as belief representations.

This paper proceeds as follows: Section 2 provides

an overview of previous online POMDP approaches. Section 3 provides a brief introduction to POMDPs and Monte Carlo tree search. Section 4 presents several algorithms for solving POMDPs on continuous spaces, discusses theoretical and practical aspects of their behavior, and finally presents the POMCPOW algorithm as a solution. Section 5 then gives experimental validation of the new algorithm.

## 2 Prior Work

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 (Kaelbling, Littman, and Cassandra 1998), although solutions to larger problems were also explored using Monte Carlo simulation and interpolation between belief states (Thrun 1999). 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, (Kurniawati, Hsu, and Lee 2008). Offline solutions for problems with continuous state and observation spaces have also been proposed (Bai, Hsu, and Lee 2014; Brechtel, Gindele, and Dillmann 2013).

There are also various solution approaches that are applicable to specific classes of POMDPs, including continuous problems. For example, Platt et al. (2010) 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 (2016) solve a monitoring problem with continuous spaces with a Gaussian process belief update. Hoey and Poupart (2005) 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 (Melchior and Simmons 2007; Prentice and Roy 2009; Bry and Roy 2011). In particular, Agha-Mohammadi, Chakravorty, and Amato (2011) present a method to take advantage of the existence of a stabilizing controller in belief space planning. Van Den Berg, Patil, and Alterovitz (2012) perform local optimization with respect to uncertainty on a precomputed path, and Indelman, Carlone, and Dellaert (2015) 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 (Ross et al. 2008). The introduction of POMCP (Silver and Veness 2010) 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 (Somani et al. 2013). Adaptive belief tree (ABT) was designed specifically to accommodate changes in the environment without having to replan from scratch (Kurniawati and Yadav 2016).

These methods can all easily handle continuous state spaces (Goldhoorn et al. 2014), 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 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 (Seiler, Kurniawati, and Singh 2015). Continuous observation Monte Carlo tree search (COMCTS) constructs observation classification trees to automatically partition the observation space in a POMCP-like approach (Pas 2012).

Although research has yielded effective solution techniques for many classes of problems, there remains a need for a simple, general purpose online POMDP solver that can handle continuous spaces, especially continuous observation spaces.

## 3 Background

This section reviews mathematical formulations for sequential decision problems and some existing solution approaches. The discussion assumes familiarity with Markov decision processes (Kochenderfer 2015), particle filtering (Thrun, Burgard, and Fox 2005), and Monte Carlo tree search (Browne et al. 2012), but reviews some details for clarity.

### 3.1 POMDPs

The Markov decision process (MDP) and partially observable Markov decision process (POMDP) can represent a wide range of sequential decision making problems. In a Markov decision process, an agent takes actions that affect the state of the system and seeks to maximize the expected value of the rewards it collects (Kochenderfer 2015). Formally, an MDP is defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the transition model, $\mathcal{R}$ is the reward function, and $\gamma$ is the discount factor. The transition model can be encoded as a set of probabilities, specifically $\mathcal{T}(s' \mid s, a)$ denotes the probability that the system will transition to state $s'$ given that action $a$ is taken in state $s$. In continuous problems, $\mathcal{T}$ is defined by probability density functions.

In a POMDP, 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 $\gamma$ 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 about the state known to the agent, $b_0$. 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, the probability of each state 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 constructing a policy $\pi$ such that, when $a_t = \pi(b_t)$, the cumulative reward or "value function" is maximized for the POMDP. Given the POMDP model, each subsequent belief can be calculated using Bayes' rule (Kaelbling, Littman, and Cassandra 1998; Kochenderfer 2015). However, the exact update is computationally intensive, so approximate approaches such as particle filtering are usually used in practice (Thrun, Burgard, and Fox 2005).

**Generative Models** For 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 MDPs** 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.

### 3.2 MCTS with Double Progressive Widening

Monte Carlo Tree Search (MCTS) is an effective and widely studied algorithm for online decision-making. It works by incrementally creating a policy tree consisting of alternating layers of state nodes and action nodes using a generative model $G$ and estimating the state-action value function, $Q(s, a)$, at each of the action nodes. The Upper Confidence Tree (UCT) version expands the tree by selecting nodes that maximize the upper confidence bound

$$UCB(s, a) = Q(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}} \qquad (1)$$

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 (Browne et al. 2012).

**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 $\lim_{N(s,a) \to 0} UCB(s, a) = \infty$, 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 $kN^\alpha$ where $N$ is the number of times the node has been visited and $k$ and $\alpha$ are hyper-parameters (Couëtoux et al. 2011). 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 (Browne et al. 2012). 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 $kN^\alpha$, 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.

### 3.3 POMCP

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 (Ross et al. 2008). 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 (Silver and Veness 2010). 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 Algorithms

This section presents several variants of the POMCP algorithm, comments on their behavior, and concludes by introducing the POMCPOW algorithm.

**Listing 1** Common procedures

```
1: procedure PLAN(b)
2:     for i ∈ 1 : n do
3:         s ← sample from b
4:         SIMULATE(s, b, d_max)
5:     return arg max_a Q(ba)
6: procedure ACTIONPROGWIDEN(h)
7:     if |C(h)| ≤ k_a N(h)^α_a then
8:         a ← NEXTACTION(h)
9:         C(h) ← C(h) ∪ {a}
10:    return arg max_{a∈C(h)} Q(ha) + c sqrt(log N(h) / N(ha))
```

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. 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 $d_{\text{max}}$ the maximum depth; $C$ is a list of the children of a node; $N$ 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.

## 4.1 POMCP-DPW

The first new algorithm that we consider is POMCP with double progressive widening (POMCP-DPW). In this algorithm, listed in Algorithm 2, the number of new children sampled from any node in the tree is limited by DPW using the parameters $k_a$, $\alpha_a$, $k_o$, and $\alpha_o$. In the case where the simulated observation is rejected (line 12), the tree search is continued with an observation selected in proportion to the number of times, $M$, it has been previously simulated (line 13) and a state is sampled from the associated belief (line 14).

This algorithm obtained remarkably good solutions for a very large autonomous freeway driving POMDP with multiple vehicles (up to 40 continuous fully observable states and 72 continuous correlated partially observable states) (Sunberg, Ho, and Kochenderfer 2017). To our knowledge, this 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 clearly 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 (Littman, Cassandra, and Kaelbling 1995; Kochenderfer 2015). 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 the supplemental material.

**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,*

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

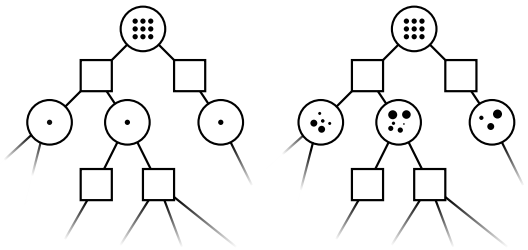
*exponentially surely in $n$, for every action $a$.*

A proof of this theorem that leverages work by Auger, Couetoux, and Teytaud (2013) is given in the supplementary material, but we provide a brief justification here. The key is that belief nodes will contain only a single state particle (see Fig. 2). 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 2), 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 (2013)). 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 (Littman, Cassandra, and Kaelbling 1995; Kochenderfer 2015).

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 QMDP-like behavior. The experiments in Section 5 show this for POMCP-DPW and DESPOT, and this is presumably the case for ABT as well.



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

Figure 2: 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.

## 4.2 Particle Filter Tree

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 approach will be referred to as the particle filter tree approach (PFT).

This algorithm has two shortcomings. First, though the belief representation has $m$ particles instead of the single particle in POMCP-DPW, it is static after the first time the node is reached, and does not grow with more simulations. Since no new particles are added to the beliefs, there is a finite probability that a valuable state will never be included in the tree, so any universal guarantees of asymptotic optimality are precluded. Second, if computational power is limited, the user must explicitly decide how much computation to devote to updates ($m$) versus decision making ($n$).

[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.

**Algorithm 2** POMCP-DPW

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

## 4.3 POMCPOW

In order to address the suboptimality of POMCP-DPW and the shortcomings of particle filter trees, 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 13), and a new state is sampled from that belief (line 15). A simple illustration of the tree is shown in Figure 2 to contrast with a POMCP-DPW tree. Because of the resampling, which can be implemented efficiently using binary search, the computational complexity is $\mathcal{O}(nd \log(n))$.

Unlike particle filter trees, since the beliefs are constantly being improved, there is hope that POMCPOW may have asymptotic convergence properties, though detailed analysis is left for the future.

It is important to note that, while POMCP, POMCP-DPW, and DESPOT only require a generative model of the problem, both POMCPOW and particle filter belief trees require a way to query the relative likelihood of different observations ($\mathcal{Z}$ in line 14). 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. For example, the authors of the original paper describing POMCP (Silver and Veness 2010) use heuristic particle reinvigoration in lieu of an observation model and importance sampling.

**Algorithm 3** POMCPOW

**procedure** SIMULATE($s$, $h$, $d$)
  **if** $d = 0$ **then**
    **return** $0$
  $a \leftarrow$ ACTIONPROGWIDEN($h$)
  $s', o, r \leftarrow G(s, a)$
  **if** $|C(ha)| \leq k_o N(ha)^{\alpha_o}$ **then**
    $M(hao) \leftarrow M(hao) + 1$
    **if** $o \notin C(ha)$ **then**
      $C(ha) \leftarrow C(ha) \cup \{o\}$
      **return** $r + \gamma$ROLLOUT$(s', hao, d)$
  **else**
    $o \leftarrow$ select $o \in C(ha)$ w.p. $\frac{M(hao)}{\sum_o M(hao)}$
  append $s'$ to $B(hao)$
  append $\mathcal{Z}(o \mid s, a, s')$ to $W(hao)$
  $s' \leftarrow$ select $B(hao)[i]$ w.p. $\frac{W(hao)[i]}{\sum_{j=1}^{m} W(hao)[j]}$
  $r \leftarrow R(s, a, s')$
  $total \leftarrow r + \gamma$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** $total$

# 5 Experiments

Numerical simulation experiments were conducted to evaluate POMCPOW's performance compared to other solvers. The set of experiments contains cases where POMCPOW outperforms all other solvers, and where it performs well, but is not the best.

## 5.1 Light Dark

In the Light Dark domain, 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 1 shows the mean reward from $500$ simulations for each solver. 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 improved solvers like POMCPOW perform better. In this case, discretization allows POMCP to outperform QMDP, but in subsequent problems where the observation space has more dimensions, discretization will not provide the same performance improvement.

## 5.2 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 and 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 a choice of 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 reward of 100 is received; If the target reaches the goal edge without being engaged, the problem ends.

An observation consists of 8 sonar beams ($\mathcal{O} = \mathbb{R}^8$) 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 agent must decide whether to use active sonar to gain information with a cost. 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 probability of success when engaging it drops to $60\%$.

Table 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 cost. The PFT approach clearly outperforms the other solvers in this domain. On problems with very simple transition models and a single information gathering decision, such as this one, PFT performs well because it is not expensive to run a full particle filter throughout the tree, but when state simulations are more expensive (as in Section 5.3), PFT is not as effective. The most important thing to note is that most of the solvers perform the same as QMDP. Only PFT and POMCPOW are able to choose to use the active sonar.

Table 1: Experimental Results

| | POMCPOW | QMDP | POMCP-DPW | DESPOT | PFT | POMCP$^{D}$ | DESPOT$^{D}$ |
|---|---|---|---|---|---|---|---|
| Light Dark | $\mathbf{62.5 \pm 0.7}$ | $2.5 \pm 1.7$ | $2.8 \pm 1.7$ | $2.4 \pm 1.7$ | $58.2 \pm 0.5$ | $30.1 \pm 1.1$ | $56.3 \pm 1.6$ |
| Sub Hunt | $45.5 \pm 1.5$ | $27.9 \pm 1.3$ | $27.8 \pm 1.3$ | $26.7 \pm 1.3$ | $\mathbf{79.0 \pm 1.1}$ | $28.2 \pm 1.3$ | $27.1 \pm 1.3$ |
| VDP Tag | $\mathbf{38.1 \pm 0.8}$ | | $-18.9 \pm 0.2$ | | $32.9 \pm 0.9$ | $-18.5 \pm 0.2$ | $-16.2 \pm 0.4$ |

The maximum scores for each problem are highlighted with bold text. Solvers with a superscript D were run on a version of the problem with a discretized action and observation spaces, but they interacted with continuous simulations of the problem.

### 5.3 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 ($\mathcal{A} = \mathbb{R} \times \{0, 1\}$). 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 ($\sigma = 0.1$). 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, moving according to 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.

Table 1 shows the mean reward for 1000 simulations of this problem for each solver. Compared to the Sub Hunt problem, the fully observable task of choosing a constant-speed course to tag the target is more difficult, so POMCPOW's ability to construct a larger search tree than PFT becomes an advantage. Numerical integration in the state transition compounds this advantage because these calculations take a larger portion of the processing time.

## 6 Conclusion

In this paper, we have proposed a new general-purpose online POMDP algorithm that is able to solve problems with continuous state, action, and observation spaces. This is a qualitative advance in capability over previous solution techniques, with the only major new requirement being explicit knowledge of the observation distribution.

This study has yielded several insights into the behavior of tree search algorithms for POMDPs. We explained why POMCP-DPW and other solvers are unable to choose costly information-gathering actions in continuous spaces and showed through experiments that, although the particle filter tree approach excels at information-gathering, it can struggle when the system dynamics are more complex. POMCPOW is able to balance both of these tasks. The computational experiments carried out for this work used only small toy problems (though they are quite large compared to many of the POMDPs previously studied in the literature), but other recent research (Sunberg, Ho, and Kochenderfer 2017) shows that POMCP-DPW is effective in very large and complex realistic domains and thus provides clear evidence the POMCPOW will also be successful.

Further work will study the theoretical properties of the algorithm. In addition, we will study better ways for choosing continuous actions. The techniques that others have studied for handling continuous actions such as generalized pattern search (Seiler, Kurniawati, and Singh 2015) and hierarchical optimistic optimization (Mansley, Weinstein, and Littman 2011) are complimentary to this work, and the combination of the two approaches will likely yield powerful tools for solving real problems.

**Acknowledgements** Toyota Research Institute ("TRI") provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

The authors would also like to thank Zongzhang Zhang for his especially helpful comments.

## References

[Agha-Mohammadi, Chakravorty, and Amato 2011] Agha-Mohammadi, A.-A.; Chakravorty, S.; and Amato, N. M. 2011. 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)*.

[Auger, Couetoux, and Teytaud 2013] Auger, D.; Couetoux, A.; and Teytaud, O. 2013. Continuous upper confidence trees with polynomial exploration–consistency. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 194–209. Springer.

[Bai, Hsu, and Lee 2014] Bai, H.; Hsu, D.; and Lee, W. S. 2014. Integrated perception and planning in the

continuous space: A POMDP approach. *International Journal of Robotics Research* 33(9):1288–1302.

[Brechtel, Gindele, and Dillmann 2013] Brechtel, S.; Gindele, T.; and Dillmann, R. 2013. Solving continuous POMDPs: Value iteration with incremental learning of an efficient space representation. In *International Conference on Machine Learning (ICML)*, 370–378.

[Browne et al. 2012] Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

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

[Couëtoux et al. 2011] Couëtoux, A.; Hoock, J.-B.; Sokolovska, N.; Teytaud, O.; and Bonnard, N. 2011. Continuous upper confidence trees. In *Learning and Intelligent Optimization*.

[Goldhoorn et al. 2014] Goldhoorn, A.; Garrell, A.; Alquézar, R.; and Sanfeliu, A. 2014. Continuous real time POMCP to find-and-follow people by a humanoid service robot. In *IEEE-RAS International Conference on Humanoid Robots*.

[Hoey and Poupart 2005] Hoey, J., and Poupart, P. 2005. Solving POMDPs with continuous or large discrete observation spaces. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1332–1338.

[Indelman, Carlone, and Dellaert 2015] Indelman, V.; Carlone, L.; and Dellaert, F. 2015. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *International Journal of Robotics Research* 34(7):849–882.

[Kaelbling, Littman, and Cassandra 1998] Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

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

[Kurniawati and Yadav 2016] Kurniawati, H., and Yadav, V. 2016. An online POMDP solver for uncertainty planning in dynamic environment. In *Robotics Research*. Springer. 611–629.

[Kurniawati, Hsu, and Lee 2008] 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*.

[Littman, Cassandra, and Kaelbling 1995] 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)*.

[Mansley, Weinstein, and Littman 2011] Mansley, C. R.; Weinstein, A.; and Littman, M. L. 2011. Sample-based planning for continuous action Markov decision processes. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

[Melchior and Simmons 2007] Melchior, N. A., and Simmons, R. 2007. Particle RRT for path planning with uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[Morere, Marchant, and Ramos 2016] Morere, P.; Marchant, R.; and Ramos, F. 2016. Bayesian optimisation for solving continuous state-action-observation POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*.

[Pas 2012] Pas, A. 2012. Simulation based planning for partially observable markov decision processes with continuous observation spaces. Master's thesis, Maastricht University.

[Platt et al. 2010] Platt, Jr., R.; Tedrake, R.; Kaelbling, L.; and Lozano-Perez, T. 2010. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*.

[Prentice and Roy 2009] Prentice, S., and Roy, N. 2009. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research* 28(11-12):1448–1465.

[Ross et al. 2008] Ross, S.; Pineau, J.; Paquet, S.; and Chaib-Draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32:663–704.

[Seiler, Kurniawati, and Singh 2015] Seiler, K. M.; Kurniawati, H.; and Singh, S. P. N. 2015. An online and approximate solver for POMDPs with continuous action space. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2290–2297.

[Silver and Veness 2010] Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*.

[Somani et al. 2013] Somani, A.; Ye, N.; Hsu, D.; and Lee, W. S. 2013. DESPOT: Online POMDP planning with regularization. In *Advances in Neural Information Processing Systems (NIPS)*, 1772–1780.

[Sunberg, Ho, and Kochenderfer 2017] Sunberg, Z. N.; Ho, C. J.; and Kochenderfer, Mykel, J. 2017. The value of inferring the internal state of traffic participants for autonomous freeway driving. In *American Control Conference (ACC)*.

[Thrun, Burgard, and Fox 2005] Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. MIT Press.

[Thrun 1999] Thrun, S. 1999. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, 1064–1070.

[Van Den Berg, Patil, and Alterovitz 2012] Van Den Berg, J.; Patil, S.; and Alterovitz, R. 2012. Motion planning under uncertainty using iterative local optimization in belief space. *International Journal of Robotics Research* 31(11):1263–1278.