Online Planning for Constrained POMDPs with Continuous Spaces through Dual Ascent

Arec Jamgochian, Anthony Corso, Mykel J. Kochenderfer
Stanford University, Department of Aeronautics and Astronautics
{arec, acorso, mykel}@stanford.edu

Abstract
Rather than augmenting rewards with penalties for undesired behavior, Constrained Partially Observable Markov Decision Processes (CPOMDPs) plan safely by imposing inviolable hard constraint value budgets. Previous work performing online planning for CPOMDPs has only been applied to discrete action and observation spaces. In this work, we propose algorithms for online CPOMDP planning for continuous state, action, and observation spaces by combining dual ascent with progressive widening. We empirically compare the effectiveness of our proposed algorithms on continuous CPOMDPs that model both toy and real-world safety-critical problems. Additionally, we compare against the use of online solvers for continuous unconstrained POMDPs that scalarize cost constraints into rewards and highlight the limitations of the default exploration scheme.

1 Introduction
Partially observable Markov decision processes (POMDPs) provide a mathematical framework for planning under uncertainty (Sondik 1978; Kochenderfer, Wheeler, and Wray 2022). An optimal policy for a POMDP maximizes the long-term expected reward that an agent accumulates while considering uncertainty from the agent’s state and dynamics. Planning, however, is often multi-objective, as agents will typically trade-off between maximizing multiple rewards and minimizing multiple costs. Though this multi-objectivity can be handled explicitly (Roijers et al. 2013), often times, multiple objectives are scalarized into a single reward function and penalties are captured through soft constraints. The drawback to this approach, however, is the need to define the parameters that weight the rewards and costs.

Constrained POMDPs (CPOMDPs) model penalties through hard constraints on expected cost-value, but are often harder to solve than POMDPs with scalarized reward functions. Policies for CPOMDPs with discrete state, action, and observation spaces can be generated offline through point-based value iteration (Kim et al. 2011), with locally-approximate linear programming (Poupart et al. 2015), or projected gradient ascent on finite-state controllers (Wray and Czuprynski 2022). Additional work develops an online receding horizon controller for CPOMDPs with large state spaces by combining Monte Carlo Tree Search (MCTS) (Silver and Veness 2010) with dual ascent to guarantee constraint satisfaction (Lee et al. 2018). However, this method is limited to discrete state and action spaces.

In this work, we extend MCTS with dual ascent to algorithms that leverage double progressive widening (Sunberg and Kochenderfer 2018; Couëtoux et al. 2011) in order to develop online solvers for CPOMDPs with large or continuous state, action, and observation spaces. Specifically we extend three continuous POMDP solvers (POMCP-DPW, POMCPOW and PFT-DPW) (Sunberg and Kochenderfer 2018) to create the constrained versions (CPOMCP-DPW, CPOMCPOW, and CPFT-DPW). In our experiments, we a) demonstrate our three solvers on toy and real-world constrained problems, b) compare against an unconstrained solver using reward scalarization, and c) demonstrate shortcomings of our solvers that are associated with pessimistic backpropagation.

2 Background
POMDPs Formally, a POMDP is defined by the 7-tuple \((S, A, O, T, Z, R, \gamma)\) consisting respectively of state, action, and observation spaces, a transition model mapping states and actions to a distribution over resultant states, an observation model mapping an underlying transition to a distribution over emitted observations, a reward function mapping an underlying state transition to an instantaneous reward, and a discount factor. An agent policy maps an initial state distribution and a history of actions and observations to an instantaneous action. An optimal policy acts to maximize expected discounted reward (Sondik 1978; Kochenderfer, Wheeler, and Wray 2022).

Offline POMDP planning algorithms (Spaan and Vlassis 2005; Kurniawati, Hsu, and Lee 2008) yield compact policies that act from any history but are typically limited to relatively small state, action, and observation spaces. Online algorithms yield good actions from immediate histories during execution (Ross et al. 2008). Silver and Veness (2010) apply Monte-Carlo Tree Search (MCTS) over histories to plan online in POMDPs with large state spaces (POMCP). Progressive widening is a technique for slowly expanding the number of children in a search tree when the multiplicity of possible child nodes is high or infinite (i.e. in continuous spaces) (Couëtoux et al. 2011; Chaslot et al. 2008).
Sunberg and Kochenderfer (2018) apply double progressive widening (DPW) to extend POMCP planning to large and continuous action and observation spaces. Additional work considers methods for selecting new actions when progressively widening using a space-filling metric (Lim, Tomlin, and Sunberg 2021) or Expected Improvement exploration (Mern et al. 2021). Wu et al. (2021) improve performance by merging similar observation branches.

**Constrained planning** Constrained POMDPs augment the POMDP tuple with a cost function $C$ that maps each state transition to a vector of instantaneous costs, and a cost budget vector $\mathbf{c}$ that the expected discounted cost returns must satisfy. An optimal CPOMDP policy $\pi$ maximizes expected discounted reward subject to hard cost constraints:

$$\max_{\pi} V^\pi_R(b_0) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(b_t, a_t) | b_0 \right]$$

subject to:

$$\sum_{t=0}^{\infty} \gamma^t C_k(b_t, a_t) | b_0 \leq c_k \ \forall k,$$

where $b_0$ is the initial state distribution, and belief-based reward and cost functions return the expected reward and costs from transitions from states in those beliefs.

Altman (1999) overviews fully-observable Constrained Markov Decision Processes (CMDPs), while Piunovskiy and Mao (2000) solve them offline using dynamic programming on a state space augmented with a constraint admissibility heuristic. Early offline CPOMDP solvers use an $\alpha$-vector formulation for value and perform cost and reward backups (Isom, Meyn, and Braatz 2008) or augment the state space with cost-to-go and perform point-based value iteration (Kim et al. 2011). CALP (Poupart et al. 2015) iterates solving a linear program with a fixed set of reachable beliefs to perform locally-approximate value iteration and expanding the belief set to represent a good policy. Walraven and Spaan (2018) improve upon this by leveraging column generation. Wray and Czuprynski (2022) represent a CPOMDP policy with a finite state controller learned offline through projected gradient ascent.

To generate good actions online, Undurti and How (2010) perform look-ahead search up to a fixed depth while using a conservative constraint-minimizing policy learned offline to estimate the cost at leaf nodes and prune unsafe branches. More recently, CC-POMCP performs CPOMDP planning by combining POMCP with dual ascent (Lee et al. 2018). Besides tracking cost values, CC-POMCP maintains estimates for Lagrange multipliers that are used to guide search. Between search queries, CC-POMCP updates Lagrange multipliers using constraint violations. CC-POMCP outperforms CALP in large state spaces.

### 3 Approach

The Lagrangian of the constrained POMDP planning problem can be formulated as

$$\max_{\pi} \min_{\lambda \geq 0} [V^\pi_R(b_0) - \lambda^\top (V^\pi_C(b_0) - \mathbf{c})].$$

**Listing 1: Common procedures**

```
1: procedure PLAN(b)
2:     \lambda \leftarrow \lambda_0
3:     for i \in 1 : n
4:         s \leftarrow \text{sample from } b
5:         SIMULATE(s, b, d_{\text{max}})
6:     a \sim \text{GREEDYPOLICY}(b, 0, 0)
7:     \lambda \leftarrow [\lambda + \alpha_i (Q_C(ba) - c)]^\dagger
8:     return \text{GREEDYPOLICY}(b, 0, \nu)
9: procedure GREEDYPOLICY(h, k, \nu)
10:     Q_A(ba) := Q(ba) - \lambda^\top Q_C(ba) + \kappa \sqrt{\log N |h| / N(ha)}
11:     \alpha \leftarrow \text{STOCHASTICPOLICY}(Q_A, \nu)
12: procedure ACTIONPROGWIDEN(h)
13:     if |C(h)| \leq k_n N(h)^{a n}
14:         a \leftarrow \text{NEXTACTION}(h)
15:     C(h) \leftarrow C(h) \cup \{a\}
16:     \pi \leftarrow \text{GREEDYPOLICY}(h, c, \nu)
17:     return \text{sample from } \pi
```

CC-POMCP (Lee et al. 2018) optimizes this objective directly by interleaving optimization for $\pi$ using POMCP and optimization of $\lambda$ using dual ascent. The planning procedure for CC-POMCP is summarized in the first procedure in Listing 1, with lines 6 and 7 depicting the dual ascent phase with an update schedule $\alpha_i$. During policy optimization, actions are always chosen with respect to the value of the Lagrangian with exploration parameter $\kappa$ (lines 10 and 11). The StochasticPolicy procedure (not shown) builds a stochastic policy of actions that are $\nu$-close to the maximal Lagrangian action-value estimate.

One significant shortcoming of using POMCP for policy optimization is that it only admits discrete action and observation spaces. In this work, we address this shortcoming using progressive widening, in which a new child node is added to parent node $p$ when the number of children nodes $|C(p)| \leq k N(p)^{a n}$, where $N(p)$ is the number of total visits to $p$, and $k$ and $\alpha$ are tree-shaping hyperparameters that can differ for action and observation branching.

Progressively widening can be straightforwardly applied to CC-POMCP. The extended version of this paper\footnote{Available at arxiv.org/abs/2212.12154} includes an outline of the resulting algorithm, CPOMC-CP. However, as noted by Sunberg and Kochenderfer (2018), progressively widening in large observation spaces leads to particle collapse as each observation node after the first step only holds a single state. To alleviate this, they propose POMCP to iteratively build up particle beliefs at each observation node and PFT-DPW to perform belief tree search using particle filter beliefs. They discuss when these algorithms may outperform vanilla progressive widening and note the absence of optimality guarantees.

In Algorithms 1 and 2, these approaches for POMCP planning in continuous spaces are combined with dual ascent in order to perform CPOMCP planning in continuous spaces. We note that these algorithms are amenable to meth-


Algorithm 1: CPOMCPOW

1: procedure SIMULATE(s, h, d)
2:   if d = 0
3:     return 0
4:   a ← ACTIONPROGWIDEN(h)
5:   s′, o, r, c ← G(s, a)
6:   if |C(ha)| ≤ k, N(ha)αω
7:     M(hao) ← M(hao) + 1
8:   else
9:     o ← select o ∈ C(ha) w.p. M(hao)/M(hao)
10:    append s′ to B(hao)
11:    append Z(o | s, a, s′) to W(hao)
12:    if o /∈ C(ha) −→ new node
13:      C(ha) ← C(ha) ∪ {a}
14:      V′, C′ ← ROLLOUT(s′, hao, d − 1)
15:    else
16:      s′ ← select B(hao)[i] w.p. W(hao)[i] / ∑j W(hao)[j]
17:      r ← R(s, a, s′)
18:      c ← C(s, a, s′)
19:      V′, C′ ← SIMULATE(s′, hao, d − 1)
20:     V ← r + γV′
21:     C ← c + γC′
22:     N(h) ← N(h) + 1
23:     N(ha) ← N(ha) + 1
24:     Q(ha) ← Q(ha) + V − Q(ha)
25:     Qc(ha) ← Qc(ha) + V − Q(ha)
26:     ¯c(ha) ← ¯c(ha) + V − Q(ha)
27: return V, C

ods that combine similar observation nodes (Wu et al. 2021) or implement better choices for actions (Mern et al. 2021) to provide optimality guarantees (Lim, Tomlin, and Sunberg 2021).

4 Experiments

We consider three constrained variants of continuous POMDP planning problems in order to empirically show the efficacy of our methods. We demonstrate the different continuous CPOMDP solvers, compare the use of CPOMCPOW against POMCPow with scalarized costs, and investigate cost backpropagation to highlight limitations. We use Julia 1.6 and the POMDPs.jl framework in our experiments (Egorov et al. 2017). Our full experimentation details, including hyperparameter choices, are available at github.com/sisl/CPOMDPExperiments.

CPOMDP Problems We enumerate the CPOMDP problems below, along with whether their state, action, and observation spaces are (D)iscrete or (C)ontinuous.

1. Constrained LightDark (C, D, C): In this adaptation of LightDark (Sunberg and Kochenderfer 2018), the agent can choose to move in discrete steps of \( \mathcal{A} = \{0, \pm 1, \pm 3, \pm 10\} \) in order to navigate to \( s \in [-1, 1] \), take action 0, and receive +100 reward. An action of 0 elsewhere accrues a −100 reward and the agent yields a per-step reward of −1. The agent starts in the dark region, \( b_0 = \mathcal{N}(2, 2) \), and can navigate towards the light region at \( s = 10 \) to help localize itself. However, there is a cliff at \( s = 12 \), above which the agent will receive a per-step cost of 1. The agent must maintain a cost budget of \( \bar{c} = 0.1 \), and so taking the +10 action immediately would violate the constraint.

2. Constrained Van Der Pol Tag (C, C, C): In this problem, a constant velocity agent must choose its orientation in order to intercept a partially observable target whose dynamics follow the Van der Pol oscillator (Sunberg and Kochenderfer 2018). In our adaptation, rather than penalizing taking good observations in the reward function, we formulate a cost constraint that dictates that the discounted number of good observations taken must be less than 2.5.

3. Constrained Spillpoint (C, C, C) This CPOMDP models safe geological carbon capture and sequestration around uncertain subsurface geometries and properties (Corso et al. 2022). In the original POMDP, instances of \( \text{CO}_2 \) leaking through faults in the geometry are heavily penalized, both for the presence of a leak and for the total amount leaked. In our adaptation, we instead impose a hard constraint of no leaking.

CPOMDP algorithm comparison Table 1 demonstrates the use of the different algorithms, comparing mean rewards and costs on the three target problems. We note that performance is highly dependent on the rollout policies, which are different for each solver and problem. Crucially, all of our methods can generate desirable behavior while satisfying hard constraints without scalarization. This is especially evident in the Spillpoint problem, where setting hard constraints minimizes \( \text{CO}_2 \) leakage while improving the reward.
generated by unconstrained POMCPOW reported by Corso et al. (2022).

**Hard constraints vs. reward scalarization** Next, we demonstrate the benefit of imposing hard constraints. To do so, we create an unconstrained LightDark problem by scalarizing the reward function and having an unconstrained solver optimize $R(s, a) = R(s, a) - \lambda C(s, a)$. We then vary choices of $\lambda$ and compare the reward and cost outcomes when using POMCPOW to solve the scalarized problem against using CPOMCPOW with the underlying CPOMDP.

In Figure 1, we depict the Pareto frontier when simulating 100 episodes at each design point. We see that the solution to the constrained problem using CPOMCPOW lies on the approximate frontier while consistently satisfying the cost constraint. We can therefore see that constrained solvers can yield high reward values at a fixed cost while eliminating the need to search over scalarization parameters.

**Cost backpropagation** Finally, we notice that using a single dual parameter to guide the search globally can result in overly conservative policies, as a globally constraining dual parameter would still guide the search in safe subtreets. To examine this, we simulate CPOMCPOW searches from the initial LightDark belief and compare backpropagating costs normally against backpropagating the minimal cost-value across sibling branches, i.e. the best-case cost assuming a closed loop. In Table 2, we compare statistics averaged across 50 searches for taking actions 1, 5, and 10. While in the unconstrained problem, the agent chooses the 10 action to localize itself quickly, we note that the constrained agent should choose the 5 action to carefully move towards the light region without overshooting and violating the cost constraint. We see that with the default search mechanism, the costs at the top level of the search tree are overly pessimistic, noting that actions of 1 or 5 should have zero cost-value as they are recoverable. For this search, propagating minimal costs achieves the desired result.

![Figure 1: The $V_R$ vs. $V_C$ Pareto frontier of solutions to the scalarized LightDark POMDP solved with POMCPOW and the Constrained LightDark CPOMDP solution. Error bars depict standard error after 100 simulations.](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>LightDark $V_R$</th>
<th>LightDark $V_C [\leq 0.1]$</th>
<th>Van Der Pol Tag $V_R$</th>
<th>Van Der Pol Tag $V_C [\leq 2.5]$</th>
<th>Spillpoint $V_R$</th>
<th>Spillpoint $V_C [\leq 0]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPOMCPOW</td>
<td>17.1±0.7</td>
<td>0.090±0.002</td>
<td>24.5±0.4</td>
<td>1.57±0.001</td>
<td>3.93±0.17</td>
<td>0.001±0.000</td>
</tr>
<tr>
<td>CPFT-DPW</td>
<td>51.9±0.4</td>
<td>0.044±0.002</td>
<td>-0.6±0.3</td>
<td>1.05±0.01</td>
<td>4.19±0.15</td>
<td>0.001±0.000</td>
</tr>
<tr>
<td>CPOMCP-DPW</td>
<td>-6.5±0.4</td>
<td>0.000±0.000</td>
<td>12.5±0.5</td>
<td>1.71±0.01</td>
<td>3.17±0.18</td>
<td>0.001±0.000</td>
</tr>
</tbody>
</table>

Table 1: Continuous CPOMDP online algorithm demonstrations comparing mean discounted cumulative rewards and costs across 100 LightDark simulations, 50 Van Der Pol Tag simulations, and 10 Spillpoint simulations.

<table>
<thead>
<tr>
<th>Model</th>
<th>$N(b_0)/N(b_a)$</th>
<th>$Q_c(b_0/a)$</th>
<th>$\Delta Q_L(b_0/a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>[0.10, 0.19, 0.08]</td>
<td>[0.18, 0.32, 1.15]</td>
<td>[5.3, 5.5, 18.9]</td>
</tr>
<tr>
<td>Min</td>
<td>[0.14, 0.33, 0.18]</td>
<td>[0.00, 0.00, 0.29]</td>
<td>[4.9, 2.1, 4.3]</td>
</tr>
<tr>
<td>Uncstr.</td>
<td>[0.09, 0.30, 0.35]</td>
<td>—</td>
<td>[8.9, 6.6, 4.5]</td>
</tr>
</tbody>
</table>

Table 2: Statistics corresponding with actions 1, 5, and 10 when running CPOMCPOW with the default cost propagation, minimal cost propagation, and on the unconstrained problem. $\Delta Q_L$ denotes the gap to the best Lagrangian action-value, and the action taken most often is bolded.

5 Conclusion

Planning under uncertainty is often multi-objective. Though multiple objectives can be scalarized into a single reward function with soft constraints, CPOMDPs provide a mathematical framework for POMDP planning with hard constraints. Previous work performs online CPOMDP planning for large state spaces, but small, discrete action and observation spaces by combining MCTS with dual ascent (Lee et al. 2018). We proposed algorithms that extend this to large or continuous action and observation spaces using progressive widening, demonstrating our solvers empirically on toy and real CPOMDP problems.

**Limitations** A significant drawback of CC-POMCP is that constraint violations are only satisfied in the limit, limiting its ability to be used as an anytime planner. This is worsened when actions and observations are continuous, as the progressive widening can miss subtrees of high cost. Finally, we note the limitation of using a single $\lambda$ to guide the whole search as different belief nodes necessitate different safety considerations.
Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1656518. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is also supported by the COMET K2—Competence Centers for Excellent Technologies Programme of the Federal Ministry for Transport, Innovation and Technology (bmvi), the Federal Ministry for Digital, Business and Enterprise (bmwi), the Austrian Research Promotion Agency (FFG), the Province of Styria, and the Styrian Business Promotion Agency (SFG). The authors would also like to acknowledge the funding support from OMV. We acknowledge Zachary Sunberg, John Mern, and Kyle Wray for insightful discussions about the material.

References


