Constrained Hierarchical Monte Carlo Belief-State Planning

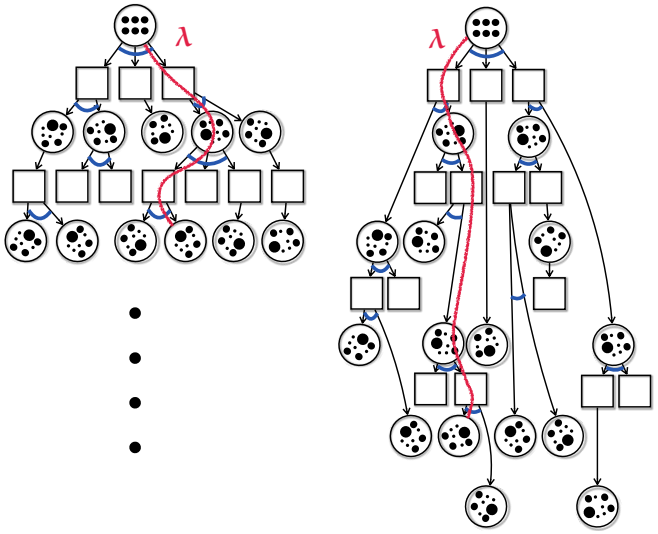
Arec Jamgochian,[1] Hugo Buurmeijer,[1] Kyle H. Wray,[1] Anthony Corso,[1] Mykel J. Kochenderfer[1]

***Abstract*— Optimal plans in Constrained Partially Observable Markov Decision Processes (CPOMDPs) maximize reward objectives while satisfying hard cost constraints, generalizing safe planning under state and transition uncertainty. Unfortunately, online CPOMDP planning is extremely difficult in large or continuous problem domains. In many large robotic domains, hierarchical decomposition can simplify planning by using tools for low-level control given high-level action primitives (options). We introduce Constrained Options Belief Tree Search (COBeTS) to leverage this hierarchy and scale online search-based CPOMDP planning to large robotic problems. We show that if primitive option controllers are defined to satisfy assigned constraint budgets, then COBeTS will satisfy constraints anytime. Otherwise, COBeTS will guide the search towards a safe sequence of option primitives, and hierarchical monitoring can be used to achieve runtime safety. We demonstrate COBeTS in several safety-critical, constrained partially observable robotic domains, showing that it can plan successfully in continuous CPOMDPs while non-hierarchical baselines cannot.**

I. Introduction

Planning in robotics requires robust regard for safety, which often necessitates careful consideration of uncertainty. Two factors contributing to uncertainty include a) the true state of the robot and surrounding environment (*state* uncertainty), and b) how that state will evolve given robot actuation (*transition* uncertainty). Constrained partially observable Markov decision processes (CPOMDPs) provide a general mathematical framework for safe planning under state and transition uncertainty by imposing constraints [1].

While offline CPOMDP planning algorithms are able to build policies for discrete environments with thousands of possible states [1], building policies in many robotic domains that are typically large or continuous necessitates online planning. Online CPOMDP planning has been scaled to large or continuous state spaces by combining Monte Carlo tree search with Lagrangian exploration and dual ascent [2] and has recently been extended to domains with continuous action and observation spaces [3]. However, search-based planning with large or continuous action and observation spaces is still extremely difficult. Common techniques try to artificially limit the width of the search tree by restricting the sets of successor nodes [4], [5]. Unfortunately, it can often be difficult to guarantee the inclusion of promising actions in the reduced search set. This problem accelerates as searches deepen, which can be especially problematic when problems require deep searches to find promising action sequences. Models can be learned for biasing action selection [6]–[8], however, this requires generating data from past experience.



(a) CPFT-DPW Tree [3] (b) COBeTS Tree

Fig. 1: In CPFT-DPW [3] (left), progressive widening (blue) is used to limit the branching factor of the Monte Carlo belief-state search tree, while dual parameters (red) are optimized to guide the search towards constraint satisfaction. COBeTS (right), leverages a hierarchy to decompose the partially observable planning problem, resulting in a search tree over options and semi-Markov belief transitions, with potentially far fewer nodes.

In many robotic planning applications, low-level controllers can be easily crafted for different high-level action primitives using domain expertise or commonly available tools (e.g. trajectory optimization). Decomposing search hierarchically over these action primitives (macro-actions, options) can *significantly* reduce the size of the search tree. Decomposition can reduce the space of actions to search over, but more importantly, reduces the search depth required to plan to the same horizon.

In this paper, we introduce Constrained Options Belief Tree Search (COBeTS), a Monte Carlo tree search algorithm that leverages hierarchical decomposition to scale online CPOMDP planning. COBeTS, depicted in Figure 1, combines the options framework to handle hierarchies [9], particle filter tree search to search over beliefs [5], progressive

[1]Stanford University, Stanford, CA 94305 USA {arec, hbuurmei, kylewray, acorso, mykel}@stanford.edu

[†]This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1656518. This work is also supported by the COMET K2—Competence Centers for Excellent Technologies Programme of the Federal Ministry for Transport, Innovation and Technology (bmvit), the Federal Ministry for Digital, Business and Enterprise (bmdw), the Austrian Research Promotion Agency (FFG), the Province of Styria, and the Styrian Business Promotion Agency (SFG).

widening to limit the number of observation *sequences* emitted from each option node, and Lagrangian exploration with dual ascent to guide the search towards safe primitives [2]. We show that if options can satisfy assigned constraint budgets, COBeTS satisfies constraints anytime. If not, dual ascent will guide the search toward safety, satisfying constraints in the limit. In our experiments, we demonstrate COBeTS on a toy domain, a carbon sequestration problem, and two robot localization problems. In each of these problems, COBeTS significantly outperforms state-of-the-art baselines that plan without hierarchical decomposition. Additionally, we demonstrate that COBeTS can satisfy constraints anytime with feasible options, and COBeTS with many options can outperform baselines because of the overall reduction in tree size induced by a hierarchy. To our knowledge, this is the first work explicitly formulating hierarchical CPOMDP planning.

In summary, our contributions are to:

- introduce COBeTS to perform online CPOMDP planning in large or continuous domains by using hierarchical decomposition,
- examine its anytime safety properties and tree complexity reduction, and
- demonstrate COBeTS on four constrained partially observable problems, including two robotic problems where non-hierarchical baselines fail to plan successfully.

II. Background

A. CPOMDPs

A CPOMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \mathbf{C}, \hat{\mathbf{c}}, \gamma)$ consisting of state, action, and observation spaces $\mathcal{S}, \mathcal{A}, \mathcal{O}$, a transition model T mapping states and actions to a distribution over resulting states, an observation model Z mapping an underlying transition to a distribution over emitted observations, a reward function R and cost function $\mathbf{C}$ mapping an underlying state transition to an instantaneous reward and vector of instantaneous, non-negative costs, a vector of cost budgets $\hat{\mathbf{c}}$, and a discount factor γ. A policy π generates actions from an initial state distribution b_0 and a history of actions $a_{0:t}$ and observations $o_{1:t}$, which together can be represented concisely as an instantaneous belief distribution over states b_t where $b_t(s) = p(s_t = s \mid b_0, a_{0:t}, o_{1:t})$. An optimal policy acts to maximize expected discounted reward while satisfying expected discounted cost budgets, that is, to optimize the following:

$$\max_{\pi} V_R^{\pi}(b_0) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(b_t, a_t) \mid b_0\right] \quad (1)$$

$$\text{s.t. } V_{C_k}^{\pi}(b_0) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t C_k(b_t, a_t) \mid b_0\right] \leq \hat{c}_k \ \forall\ k, \quad (2)$$

where belief-based reward and cost functions return the expected reward and costs from transitions from states in those beliefs [1], [10]–[13].

Offline CPOMDP planning algorithms solve for compact policies that map from any history to instantaneous actions. Offline CPOMDP solution methods include dynamic programming [13], [14], approximate linear programming [1], column generation [15], and projected gradient ascent [16]. Unfortunately, offline solutions are limited to problems with small state, action, and observation spaces.

Online CPOMDP planning algorithms can generate better solutions by searching across immediately reachable beliefs [17]. CC-POMCP [2] plans online in extremely large state spaces by performing Lagrangian-guided partially observable Monte Carlo tree search [18], with dual ascent to optimize the Lagrange multipliers. CPFT-DPW and CPOMCPOW [3] scale constrained online Monte Carlo planning to large action and observation spaces by limiting the search tree branching factor using progressive widening [4], [5].

B. Hierarchical Planning

Hierarchical planning simplifies difficult planning problems by favorably decomposing them into more tractable subproblems [19], [20]. Applications to robotics date back to the use of high-level task planning and low-level execution and runtime monitoring on the Shakey robot [21]. One common framework for planning hierarchically is through options [9], in which a primitive controller (option) $\hat{a}$ is chosen from a finite set of options $\hat{\mathcal{A}}$ and executed until termination, upon which a new valid option is chosen. A partially observable options model augments an underlying POMDP problem with the set $\{\mathcal{I}_{\hat{a}}, \pi_{\hat{a}}^L, \beta_{\hat{a}}\}_{\hat{a} \in \hat{\mathcal{A}}}$, that for every option, defines a set of belief-states $\mathcal{I}$ from which it can be initialized, a low-level control policy $\pi^L(a \mid b)$ returning actions from the underlying action space $\mathcal{A}$, and a function β that returns the probability that an option will terminate in a given belief. When beginning execution of a new option, the high-level policy π must choose from the subset of options that are available from the instantaneous belief, $\{\hat{a} \mid b_t \in \mathcal{I}_{\hat{a}}\}$. If desired, the underlying action space can be included in the set of options.

An implementation of hierarchical policy execution in CPOMDPs using the options framework is depicted in Algorithm 1. During execution, the low-level policy acts in the environment (lines 6–7), updates the cost budget with the expected instantaneous cost (line 8), and updates the state belief using a new observation (line 9). A high-level `SelectOption` policy chooses a new option whenever an executing option terminates (lines 4–5).

Related previous work performed online hierarchical planning for *unconstrained* POMDPs [22] by combining partially observable MCTS with MaxQ [23], an alternative framework for hierarchical planning. Though not explicitly using hard constraints, additional recent work has focused on safe planning in large partially observable robotic domains by using a hierarchical information roadmap to manage local risks safely on large exploration missions [24]. For problems with favorable state partitions (e.g. path planning on a grid of neighboring states), work has also been done to solve large, *fully-observable*, constrained MDPs hierarchically by combining global CMDP solutions over coarse partitions with local solutions over underlying states [25].

Algorithm 1 Hierarchical Execution in an Options CPOMDP

procedure EXECUTE($b_0, \hat{\mathbf{c}}$)
  $\hat{a} \leftarrow \emptyset, b \leftarrow b_0$
  while $\neg$TERMINAL(b)
    if $\hat{a} = \emptyset \vee$ TERMINATE($\hat{a}, b$)
      $\hat{a} \leftarrow$ SELECTOPTION($b, \hat{\mathbf{c}}$)
    $a \leftarrow$ ACTION($\hat{a}, b$)
    $o \leftarrow$ STEP(b, a)
    $\hat{\mathbf{c}} \leftarrow \left[\frac{\hat{\mathbf{c}} - \mathbf{C}(b,a)}{\gamma}\right]^+$
    $b \leftarrow$ UPDATEBELIEF(b, a, o)

III. METHODOLOGY

A. Preliminaries

We formulate hierarchical planning in a CPOMDP using the options framework. Options induce two processes: a low-level Markov process over the underlying state, action, and observation space, and a high-level semi-Markov process between successive option calls. With option policies defined a priori, the underlying model is a constrained partially observable semi-Markov decision process (CPOSMDP). Consequently, we now briefly cover the CSMDP, its generalization as CPOSMDP, and the CPOSMDP's equivalent belief-state CSMDP. More details can be found in related work such as by Vien and Toussaint [22].

A CSMDP is defined in a way similar to a CMDP, with the inclusion of a now semi-Markov transition function T that defines the joint probability of the successor state alongside the number of steps required to transition given a state and action, $p(s', \tau \mid s, a)$. Decisions are made at successive decision epochs e, each indexing a time step t_e when an action a_e begins executing and its duration τ_e where $t_{e+1} = t_e + \tau_e$.

Similarly, a CPOSMDP is defined with the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, \mathbf{C}, \hat{\mathbf{c}}, \gamma)$, where P models the joint semi-Markov transition and observation functions $p(s', \mathbf{o}, \tau \mid s, a)$, where $\mathbf{o} \in \mathcal{O}^\tau$ is the sequence of emitted observations in a τ-step semi-Markov transition. As with POMDPs and their equivalent belief-state MDP representations, we can define a CPOSMDP equivalently as a belief-state CSMDP $(\tilde{\mathcal{S}}, \mathcal{A}, \tilde{T}, \tilde{R}, \tilde{\mathbf{C}}, \hat{\mathbf{c}}, \gamma)$. The belief states are $b \in \tilde{\mathcal{S}} = \triangle(\mathcal{S})$, with transitions $\tilde{T}(b, a, b' = ba\mathbf{o}, \tau) = \sum_{s,s' \in \mathcal{S}} b(s) b'(s') P(s, a, s', \mathbf{o}, \tau)$, rewards $\tilde{R}(b, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a)$, and costs $\tilde{\mathbf{C}}(b, a) = \sum_{s \in \mathcal{S}} b(s) \mathbf{C}(s, a)$.

Proposition 1: For all policies π, the reward value functions and cost value functions of the belief-state CSMDP are equal to those of the CPOSMDP, that is, for all $b \in \tilde{\mathcal{S}}, \tilde{V}_R^\pi(b) = V_R^\pi(b)$ and $\tilde{\mathbf{V}}_C^\pi(b) = \mathbf{V}_C^\pi(b)$.

The proof follows directly from Theorem 2 of Vien and Toussaint [22], with vector costs and cost-values treated analogously as scalar rewards and reward-values in the original proof. This theoretical result lays the groundwork for COBeTS as it allows us to solve CPOSMDPs by solving their equivalent belief-state CSMDPs.

Algorithm 2 Constrained Options Belief Tree Search

1: **procedure** SELECTOPTION($b, \hat{\mathbf{c}}$)
2: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}_0$
3: **for** $i \in 1 : n$
4: $Q_\lambda(b\hat{a}) := Q(b\hat{a}) - \boldsymbol{\lambda}^\top \mathbf{Q}_C(b\hat{a})$
5: SIMULATE($b, \hat{\mathbf{c}}, d_{\max}$)
6: $\hat{a} \leftarrow \arg\max_{\hat{a}} Q_\lambda(b\hat{a})$
7: $\boldsymbol{\lambda} \leftarrow [\boldsymbol{\lambda} + \alpha_i(\mathbf{Q}_C(b\hat{a}) - \hat{\mathbf{c}})]^+$
8: **return** $\arg\max_{\hat{a}} Q(b\hat{a})$ s.t. $\mathbf{Q}_C(b\hat{a}) \leq \hat{\mathbf{c}}$
9: **procedure** OPTIONPROGWIDEN($b, \hat{\mathbf{c}}$)
10: **if** $|C(b)| \leq k_a N(b)^{\alpha_a}$
11: $\hat{a} \leftarrow$ SAMPLENEXTOPTION($b, [\hat{\mathbf{c}}]^+$)
12: $C(b) \leftarrow C(b) \cup \{\hat{a}\}$
13: $Q_{\lambda UCB}(b\hat{a}) := Q_\lambda(b\hat{a}) + \kappa \sqrt{\frac{\log N(b)}{N(b\hat{a})}}$
14: **return** $\arg\max_{\hat{a}} Q_{\lambda UCB}(b\hat{a})$
15: **procedure** SIMULATE(b, $\hat{\mathbf{c}}$, d)
16: **if** $d \leq 0$
17: **return** $0, \mathbf{0}$
18: $\hat{a} \leftarrow$ OPTIONPROGWIDEN($b, \hat{\mathbf{c}}$)
19: **if** $|C(b\hat{a})| \leq k_o N(b\hat{a})^{\alpha_o}$
20: $b', \tilde{r}, \tilde{\mathbf{c}} \leftarrow G_{\text{PF}(m)}(b, \text{ACTION}(\hat{a}, b))$
21: $\tau \leftarrow 1$
22: **while** $\neg$TERMINATE($\hat{a}, b'$) $\wedge$ $(d - \tau > 0)$
23: $b', r, \mathbf{c} \leftarrow G_{\text{PF}(m)}(b', \text{ACTION}(\hat{a}, b'))$
24: $\tilde{r} \leftarrow \tilde{r} + \gamma^\tau r$
25: $\tilde{\mathbf{c}} \leftarrow \tilde{\mathbf{c}} + \gamma^\tau \mathbf{c}$
26: $\tau \leftarrow \tau + 1$
27: $C(b\hat{a}) \leftarrow C(b\hat{a}) \cup \{(b', \tilde{r}, \tilde{\mathbf{c}}, \tau)\}$
28: $V', \mathbf{C}' \leftarrow$ ESTIMATEVALUE($b', \frac{\hat{\mathbf{c}} - \tilde{\mathbf{c}}}{\gamma^\tau}, d - \tau$)
29: **else**
30: $b', \tilde{r}, \tilde{\mathbf{c}}, \tau \leftarrow$ sample uniformly from $C(b\hat{a})$
31: $V', \mathbf{C}' \leftarrow$ SIMULATE($b', \frac{\hat{\mathbf{c}} - \tilde{\mathbf{c}}}{\gamma^\tau}, d - \tau$)
32: $V \leftarrow \tilde{r} + \gamma^\tau V'$
33: $\mathbf{C} \leftarrow \tilde{\mathbf{c}} + \gamma^\tau \mathbf{C}'$
34: $N(b) \leftarrow N(b) + 1$
35: $N(b\hat{a}) \leftarrow N(b\hat{a}) + 1$
36: $Q(b\hat{a}) \leftarrow Q(b\hat{a}) + \frac{V - Q(b\hat{a})}{N(b\hat{a})}$
37: $\mathbf{Q}_C(b\hat{a}) \leftarrow \mathbf{Q}_C(b\hat{a}) + \frac{\mathbf{C} - \mathbf{Q}_C(b\hat{a})}{N(b\hat{a})}$
38: **return** $V, \mathbf{C}$

B. Constrained Options Belief-Tree Search (COBeTS)

The idea behind COBeTS is to select new options in a large CPOSMDP by planning over the equivalent belief-state CSMDP. COBeTS augments CPFT-DPW [3], a recent algorithm for online belief-state CMDP planning, with careful consideration for the options framework. That is, rather than search over actions a, COBeTS searches over options $\hat{a}$ and samples their induced semi-Markov belief-state transitions.

The COBeTS option-selection procedure is outlined in Algorithm 2 with changes from CPFT-DPW highlighted in blue. The procedure is a recursive Monte Carlo tree search on a particle filter belief-state b. In `OptionProgWiden`, option selection (lines 13–14) is guided by a Lagrangian upper confidence bound heuristic that uses the current es-

timate of the dual parameters to trade off between reward and constraint objectives (line 4) and an exploration bonus based on visit counts. Dual parameters are updated between searches through dual ascent (line 7), in which constraint violations during search induce strengthening of the dual parameters and vice versa.

After selecting a search option (line 18), COBeTS imagines executing that option until its termination, resulting in a semi-Markov belief transition (lines 20–27). Option execution uses sampled low-level actions and observations to update the m-state particle filter belief at every step (line 23) while tracking the discounted rewards and costs accumulated along the option trajectory. New leaf nodes are initialized with value estimates that can be generated from default policy rollouts or heuristics (`EstimateValue` in line 28). The simulated reward and cost values are used to make temporal difference updates and are then backpropagated (lines 32–38).

Searching across a large set of options or resulting transitions necessitates techniques to limit the size of the tree. Progressive widening artificially limits the branching factor of a node as a function of its visit count $N(b)$, limiting the number of children to $|Ch(b)| \approx kN(b)^{\alpha}$, where $k > 0$ and $\alpha \in (0, 1)$ are hyperparameters that control the shape of the widening [4], [5]. COBeTS implements progressive widening on the option space (lines 9–14) and on the semi-Markov belief-state transition space (line 19). Different option sampling strategies in `SampleNextOption` can ensure coverage of the option space [26]. Since the transition distribution $T = p(b', \tau \mid b, \hat{a})$ is often continuous and uncountable, COBeTS benefits greatly from progressive widening on its belief transitions for the same reasons as large continuous POMDPs [5].

Hierarchical decomposition provides computational advantages that can be estimated through the ratio in sizes between analogous CMDP and CSMDP search trees. Consider searching across a belief-state CMDP with average action branching of cardinality A and state transition branching with average cardinality O alongside a belief-state CSMDP with action branching of average cardinality $c_1 A$, and state transition branching with average cardinality $c_2 O$ after an average of τ steps.

With a fixed time horizon T, searching over the CSMDP instead of the CMDP improves computational complexity by a factor of $\mathcal{O}((c_1 c_2)^{-T/\tau}(AO)^{T(\tau-1)/\tau})$, as the ratio in the computational complexity can be expressed as the ratio of the tree sizes:

$$\frac{\text{CSMDP size}}{\text{CMDP size}} = \mathcal{O}\left(\frac{(c_1 c_2 AO)^{\frac{T}{\tau}}}{(AO)^T}\right) = \mathcal{O}\left(\frac{(c_1 c_2)^{\frac{T}{\tau}}}{(AO)^{\frac{T(\tau-1)}{\tau}}}\right). \tag{3}$$

This ratio allows us to analyze the significant improvement in computational complexity induced through a hierarchical decomposition. If our hierarchical decomposition had the same action and observation branching factors ($c_1 = c_2 = 1$), it would result in a tree that is smaller by a factor of $(AO)^{\frac{T(\tau-1)}{\tau}}$. This gives significant leeway and allows us to, for example, compensate for designing a large number of many-step options.

C. Maintaining Feasibility Anytime with Options

Though combining Lagrangian Monte Carlo Tree Search with dual ascent guides search away from constraint violations in the limit, it does not guarantee anytime constraint satisfaction. When executing a hierarchical controller, runtime monitoring can be used to ensure safety online by terminating options and replanning in case of impending constraint violations. However, unsafe search could still lead to states where safe replanning is not possible. In this section, we show that when options are feasible when executed under an assigned budget for their decision epoch, COBeTS can maintain global feasibility anytime. To show this, we first define local feasibility, one-step global feasibility, and global feasibility.

Definition 1: An option $\hat{a}_e$ chosen at decision epoch e in b_e is *locally feasible* given a budget $\hat{\mathbf{c}}_e$ if $\mathbf{Q}^{\pi}_{\mathbf{C}}(b_e, \hat{a}_e) \leq \hat{\mathbf{c}}_e$.

Definition 2: An option $\hat{a}_e$ chosen at decision epoch e in b_e is *one-step globally feasible* if $\mathbf{C}_{e-1} + \mathbf{Q}^{\pi}_{\mathbf{C},e}(b_e, \hat{a}_e) \leq \hat{\mathbf{c}}$, where $\mathbf{C}_{e-1} = \sum_{e'=0}^{e-1} \gamma^{t_{e'}} \tilde{\mathbf{c}}_{e'} = \sum_{t=0}^{t_e-1} \gamma^t \mathbf{C}(b_t, a_t)$.

Definition 3: An algorithm or policy π is said to be *globally feasible* if $\mathbf{V}^{\pi}_{\mathbf{C}}(b_0) \leq \hat{\mathbf{c}}$.

Informally, a locally feasible option is guaranteed to satisfy a set cost budget while it is in control, a one-step globally feasible option can be applied once and is guaranteed to satisfy the global budget, and global feasibility states that the original constraints from Eq. (2) are satisfied.

With these definitions, Proposition 2 below shows that for a CPOSMDP, if a locally feasible option is chosen with a particular assignment of $\hat{\mathbf{c}}_e$, then it ensures that one-step is globally feasible.

Proposition 2: For policy π and locally feasible option $\hat{a}_e$ at decision epoch e with accumulated costs $\mathbf{C}_{e-1}$, if $\hat{\mathbf{c}}_e = \frac{\hat{\mathbf{c}} - \mathbf{C}_{e-1}}{\gamma^{t_e}} \geq 0$ then $\hat{a}_e$ is one-step globally feasible.

Proof: By definition of a locally feasible option and the choice of $\hat{\mathbf{c}}_e$:

$$\mathbf{Q}_{\mathbf{C}}(b_e, \hat{a}_e) \leq \hat{\mathbf{c}}_e = \frac{\hat{\mathbf{c}} - \mathbf{C}_{e-1}}{\gamma^{t_e}}$$

$$\mathbf{C}_{e-1} + \gamma^{t_e} \mathbb{E}\Big[\sum_{t=t_e}^{\infty} \gamma^{t-t_e} \mathbf{C}(b_t, a_t) \mid b_e, \hat{a}_e, \pi\Big] \leq \hat{\mathbf{c}}$$

$$\mathbf{C}_{e-1} + \mathbf{Q}^{\pi}_{\mathbf{C},e}(b_e, \hat{a}_e) \leq \hat{\mathbf{c}}.$$

Thus, by Definition 2, $\hat{a}_e$ is one-step globally feasible. ■

These results imply that COBeTS is globally feasible when its options are locally feasible, that is, when they satisfy the budgets passed to `SampleNextOption` (line 11).

Proposition 3: COBeTS is globally feasible if all its options $\hat{a}_e$ are locally feasible given COBeTS assignments of $\hat{\mathbf{c}}_e \geq 0$ for all e.

Proof: By construction. Consider any decision epoch e. As given, consider any COBeTS option $\hat{a}_e$. By definition of COBeTS, it assigns $\hat{\mathbf{c}}_e = \left[\frac{\hat{\mathbf{c}} - \mathbf{C}_{e-1}}{\gamma^{t_e}}\right]^{+} \geq 0$. By Proposition 2, it is one-step globally feasible. Since this is true for all e, COBeTS is globally feasible. ■

IV. Experiments

Our experiments consider online planning in four large safety-critical, partially observable planning problems in order to empirically demonstrate the efficacy of COBeTS. We compare COBeTS against different non-hierarchical solvers on our target domains, investigate its anytime properties, and show that it can yield better plans even when the number of options far exceeds the number of underlying actions.

We use Julia 1.9 and the POMDPs.jl framework for our experiments [27]. In the following sections, we outline the CPOMDP target problems, briefly describe their hierarchical decompositions, and discuss the main results from our experiments. For full experimentation details, including CPOMDP modeling details, the precise options crafted, and choices of hyperparameters, please refer to our code, which we have open-sourced at `github.com/sisl/COBTSExperiments`.

A. CPOMDP Problems and Option Policies

We highlight the CPOMDP problem domains used in our experiments below, whether their state, action, and observation spaces are (D)iscrete or (C)ontinuous, and provide an overview of the types of options crafted for execution.

1) Constrained LightDark [3] (C, D, C): In this one-dimensional robot localization problem adapted from LightDark [5], [28], the robot must first safely localize itself before navigating to the goal. The robot can move in discrete steps of $\mathcal{A} = \{0, \pm 1, \pm 5, \pm 10\}$ in order to navigate to $s \in [-1, 1]$, take action 0, and receive $+100$ reward, but taking action 0 elsewhere accrues a -100 reward. The robot accrues a per-step reward of -1. The agent starts in the dark region, $b_0 = \mathcal{N}(2, 2^2)$, and can navigate towards the light region at $s = 10$ to help localize itself with less noisy observations. However, the robot must avoid entering a constraint region above $s = 12$ where it will receive a per-step cost of 1 and violate a budget of $\hat{c} = 0.1$. As such, taking the $+10$ action immediately would violate the constraint in expectation.

We template four types of options for this problem. `GoToGoal` greedily navigates the robot's mean position to the goal and terminates. `LocalizeFast` greedily navigates the robot's mean position to the light region until the belief uncertainty is sufficiently small. `LocalizeFromBelow` adjusts the navigation technique for localization so that the robot's mean position does not overshoot the light region. `LocalizeSafe` uses the robot's position uncertainty while localizing to minimize the risk that the robot violates the constraint.

2) Constrained Spillpoint [3] (C, C, C): This problem models safe geological carbon capture and sequestration around uncertain subsurface geometries and properties. In the original POMDP [29], instances of CO_2 leaking through faults in the geometry are heavily penalized, both for the presence of a leak and for the total amount leaked. The constrained adaptation imposes a constraint of $\hat{c} = 1 \times 10^{-6}$ to ensure minimal CO_2 leakage.

The options for the spillpoint problem include `InferGeology` and `SafeFill`. The `InferGeology` begins by injecting 90% of the CO_2 volume of the lowest-volume instance of the geology according to the current belief. Then, a sequence of observations of conducted which provides information on the shape of the geology (these observations indicate the presence of CO_2 at various spatial locations). The `InferGeology` is templated so a variety of observation sequences can be selected. The `SafeFill` option involves injecting 90% of the CO_2 volume of the lowest-volume instance of the geology according to the current belief and then terminating the episode. The options (with five versions of `InferGeology`) were combined with the standard set of five individual actions for a total of 11 options.

3) Constrained Bumper Roomba (C, D, D): Roomba models a robot with an uncertain initial pose in a fixed environment as it uses its sensors to navigate to a goal region while avoiding a penalty region [30]. In this work, we augment the problem to include a constraint region that the robot must avoid traveling through as it navigates to the goal. States are defined by the continuous pose of the robot on the map, actions allow the robot to turn or move by discrete amounts, and while the robot does not have access to its true pose, in Bumper Roomba, it receives a binary observation when it collides with a wall.

Bumper Roomba crafts three types of options: `TurnAndGo` options turn the robot a fixed amount then navigate until the robot collides with a wall, `GreedyGoToGoal` greedily navigates to the goal using the robot's mean pose, and `SafeGoToGoal` navigates to the goal while imposing a barrier function around the constraint region.

4) Constrained Lidar Roomba (C, D, C): This CPOMDP augments Constrained Bumper Roomba with a Lidar sensor that noisily observes the distance to the nearest wall along the robot's heading, with noise proportional to the distance. Rather than using `TurnAndGo` options for localization, Constrained Lidar Roomba implements `Spin` options that turn the robot for different periods of time and with different turn radii to localize.

B. Experiments and Discussion

1) CPOMDP algorithm comparison: To evaluate COBeTS, we measure the mean discounted cumulative reward and cost for different planning episodes on the aforementioned CPOMDP domains. We benchmark COBeTS against the closest non-hierarchical online CPOMDP planning algorithms, CPOMCPOW and CPFT-DPW [3], which perform Lagrangian MCTS on the state spaces and belief-state spaces respectively. Table I summarizes the performance of the algorithms on the different CPOMDP domains, averaged across 100 LightDark, 10 Spillpoint, and 50 Roomba simulations.

In summary, we see that COBeTS is able to significantly outperform against baselines on all domains while satisfying cost constraints. In both Roomba problems, baselines are unable to search deep enough to localize and get to the goal, and instead meander while accruing step penalties to avoid the risk of violating the constraint.

Model	LightDark		Spillpoint		Bumper Roomba		Lidar Roomba	
	$\hat{V}_R$	$\hat{V}_C$ $[\leq 0.1]$	$\hat{V}_R$	$\hat{V}_C$ $[\leq 10^{-6}]$	$\hat{V}_R$	$\hat{V}_C$ $[\leq 0.1]$	$\hat{V}_R$	$\hat{V}_C$ $[\leq 0.1]$
COBeTS	**68.6**$\pm$0.7	0.027$\pm$0.015	**3.40**$\pm$0.66	0.000$\pm$0.000	**5.73**$\pm$0.67	0.038$\pm$0.038	**5.23**$\pm$0.67	0.020$\pm$0.019
CPFT-DPW	5.9$\pm$7.7	0.000$\pm$0.000	1.50$\pm$0.39	0.000$\pm$0.000	-4.76 ± 0.00	0.036$\pm$0.036	-4.57 ± 0.19	0.000$\pm$0.000
CPOMCPOW	-9.2 ± 8.0	0.032$\pm$0.015	1.51$\pm$0.40	$1.0 \cdot 10^{-5} \pm 0.9 \cdot 10^{-5}$	-3.76 ± 0.42	0.000$\pm$0.000	-4.55 ± 0.20	0.000$\pm$0.000

TABLE I: Online CPOMDP algorithm demonstrations comparing mean discounted cumulative rewards ($\hat{V}_R$) and costs ($\hat{V}_C$) across 100 LightDark simulations, 10 Spillpoint simulations, and 50 Bumper and Lidar Roomba simulations. COBeTS consistently satisfies constraints while outperforming both the CPFT-DPW baseline and the CPOMCPOW baseline.

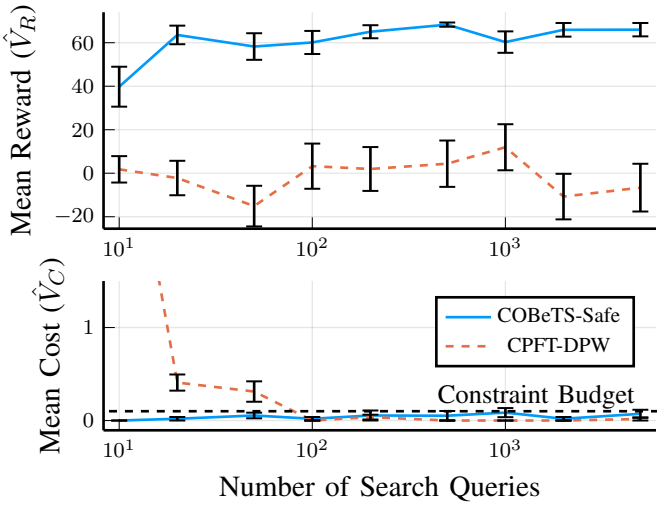


Fig. 2: Mean cumulative discounted rewards (above) and costs (below) vs. number of tree queries across 50 Constrained LightDark simulations when using COBeTS with feasible options. COBeTS stays safe anytime while CPFT-DPW only satisfies constraints in the limit.

2) Anytime constraint satisfaction: To highlight the anytime constraint satisfaction with COBeTS we vary the number of search queries in the Constrained LightDark CPOMDP with the robot restricted to `GoToGoal` and two different `LocalizeSafe` options, all of which are feasible from the initial belief. The results averaged across 50 simulations are depicted in Figure 2. We see that even with low numbers of search queries, COBeTS satisfies the constraints while achieving high reward, while CPFT-DPW only satisfies the constraints as the number of queries is increased.

3) Searching over many options: Finally, we investigate the impact that the action branching factor has on search quality in LightDark. We vary the branching factor in the COBeTS search by adding options using different strategies. COBeTS-Unc. adds options that localize to different sampled uncertainties, while COBeTS-Random adds options that execute three randomly selected non-terminal actions. The results, depicted in Figure 3, show that COBeTS still finds quality plans with a significant number of options to search over, much greater than the number of actions in the underlying problem (seven). These results support the analysis presented in Equation (3), that search complexity reduction from hierarchical decomposition can compensate for increased action branching.

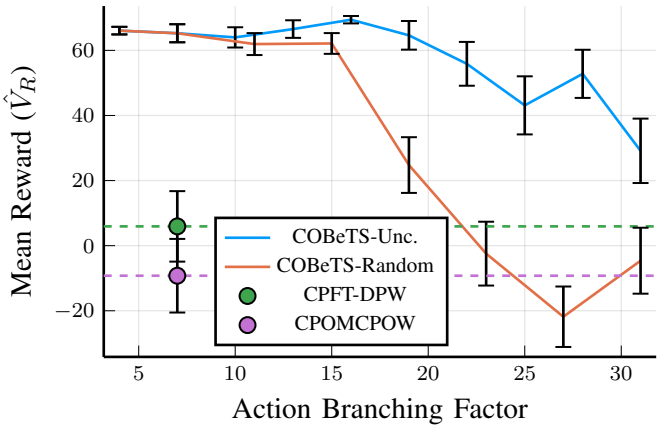


Fig. 3: Mean cumulative discounted rewards for different numbers of options averaged over 50 Constrained LightDark simulations. All costs are feasible (not shown). COBeTS can retain high reward at larger action branching factors because hierarchy induces a smaller overall tree size.

V. Conclusion

Safe robotic planning under state and transition uncertainty can be naturally expressed as a CPOMDP, but CPOMDPs are extremely difficult to solve in large problem domains. Recent works scaled online search-based CPOMDP planning to large spaces [2], [3], but with limited scope or expertly crafted heuristics. In robotics, planning can often be favorably decomposed hierarchically, separating high-level action primitives and low-level control. In this work, we introduced Constrained Options Belief Tree Search (COBeTS) to improve online CPOMDP planning when favorable hierarchies exist by performing a belief-state Monte Carlo tree search over options. We showed that COBeTS with feasible options will satisfy constraints anytime and demonstrated its success on large planning domains where recent methods fail.

Limitations and future work: A significant limitation of COBeTS is the necessity to craft low-level policies. Recent work uses language models to construct policies automatically [31] and if combined with COBeTS, could enable hierarchical constrained search to compensate for uncertainty in policy generation. A second limitation is that though COBeTS biases the search toward safety, it requires feasible options in order to satisfy constraints anytime. Future work can examine propagating cost bounds [17] or using search heuristics generated from past experience [7], [8], [32] or natural language priors [33] to achieve safety under more general conditions.

References

[1] P. Poupart, A. Malhotra, P. Pei, K.-E. Kim, B. Goh, and M. Bowling, "Approximate linear programming for constrained partially observable Markov decision processes," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 29, 2015.

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

[3] A. Jamgochian, A. Corso, and M. J. Kochenderfer, "Online planning for constrained POMDPs with continuous spaces through dual ascent," in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 33, 2023, pp. 198–202.

[4] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *International Conference on Learning and Intelligent Optimization (LION)*, Springer, 2011, pp. 433–445.

[5] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2018.

[6] J. Mern, A. Yildiz, Z. Sunberg, T. Mukerji, and M. J. Kochenderfer, "Bayesian optimized Monte Carlo planning," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, 2021, pp. 11 880–11 887.

[7] P. Cai and D. Hsu, "Closing the planning–learning loop with application to autonomous driving," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 998–1011, 2022.

[8] R. J. Moss, A. Corso, J. Caers, and M. J. Kochenderfer, "BetaZero: Belief-state planning for long-horizon POMDPs using learned approximations," 2023. arXiv: 2306.00249 [cs.AI].

[9] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[10] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.

[11] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. MIT Press, 2022.

[12] E. Altman, *Constrained Markov Decision Processes*. CRC Press, 1999, vol. 7.

[13] J. D. Isom, S. P. Meyn, and R. D. Braatz, "Piecewise linear dynamic programming for constrained POMDPs," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

[14] D. Kim, J. Lee, K.-E. Kim, and P. Poupart, "Point-based value iteration for constrained POMDPs," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[15] E. Walraven and M. T. Spaan, "Column generation algorithms for constrained POMDPs," *Journal of Artificial Intelligence Research*, vol. 62, pp. 489–533, 2018.

[16] K. H. Wray and K. Czuprynski, "Scalable gradient ascent for controllers in constrained POMDPs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9085–9091.

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

[18] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2010, pp. 2164–2172.

[19] E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artificial Intelligence*, vol. 5, no. 2, pp. 115–135, 1974.

[20] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," vol. 10, 1997.

[21] R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and executing generalized robot plans," *Artificial intelligence*, vol. 3, pp. 251–288, 1972.

[22] N. A. Vien and M. Toussaint, "Hierarchical Monte-Carlo planning," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 29, 2015.

[23] T. G. Dietterich, "The MAXQ method for hierarchical reinforcement learning.," in *International Conference on Machine Learning (ICML)*, vol. 98, 1998, pp. 118–126.

[24] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, "PLGRIM: Hierarchical value learning for large-scale exploration in unknown environments," in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 31, 2021, pp. 652–662.

[25] S. Feyzabadi and S. Carpin, "Planning using hierarchical constrained Markov decision processes," *Autonomous Robots*, vol. 41, pp. 1589–1607, 2017.

[26] M. H. Lim, C. J. Tomlin, and Z. N. Sunberg, "Voronoi progressive widening: Efficient online solvers for continuous state, action, and observation POMDPs," in *IEEE Conference on Decision and Control (CDC)*, 2021, pp. 4493–4500.

[27] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "POMDPs.jl: A framework for sequential decision making under uncertainty," *Journal of Machine Learning Research*, vol. 18, no. 26, pp. 1–5, 2017.

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

[29] A. Corso, Y. Wang, M. Zechner, J. Caers, and M. J. Kochenderfer, "A POMDP model for safe geological carbon sequestration," in *Advances in Neural Information Processing Systems (NeurIPS)*, ser. Tackling Climate Change with Machine Learning Workshop, 2022.

[30] C. Wu, G. Yang, Z. Zhang, Y. Yu, D. Li, W. Liu, and J. Hao, "Adaptive online packing-guided search for POMDPs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 28 419–28 430, 2021.

[31] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500.

[32] D. Parthasarathy, G. Kontes, A. Plinge, and C. Mutschler, "C-MCTS: Safe planning with Monte Carlo tree search," 2023. arXiv: 2305.16209 [cs.LG].

[33] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," 2023. arXiv: 2305.14078 [cs.RO].