ConstrainedZero: Chance-Constrained POMDP Planning Using Learned Probabilistic Failure Surrogates and Adaptive Safety Constraints

Robert J. Moss[1], **Arec Jamgochian**[1], **Johannes Fischer**[1,2], **Anthony Corso**[1], and **Mykel J. Kochenderfer**[1]
[1]Stanford University, Stanford, CA
[2]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{mossr, arec, acorso, mykel}@stanford.edu, johannes.fischer@kit.edu

Abstract

To plan safely in uncertain environments, agents must balance utility with safety constraints. Safe planning problems can be modeled as a chance-constrained partially observable Markov decision process (CC-POMDP) and solutions often use expensive rollouts or heuristics to estimate the optimal value and action-selection policy. This work introduces the *ConstrainedZero* policy iteration algorithm that solves CC-POMDPs in belief space by learning neural network approximations of the optimal value and policy with an additional network head that estimates the failure probability given a belief. This failure probability guides safe action selection during online Monte Carlo tree search (MCTS). To avoid overemphasizing search based on the failure estimates, we introduce Δ-MCTS, which uses adaptive conformal inference to update the failure threshold during planning. The approach is tested on a safety-critical POMDP benchmark, an aircraft collision avoidance system, and the sustainability problem of safe CO_2 storage. Results show that by separating safety constraints from the objective we can achieve a target level of safety without optimizing the balance between rewards and costs.

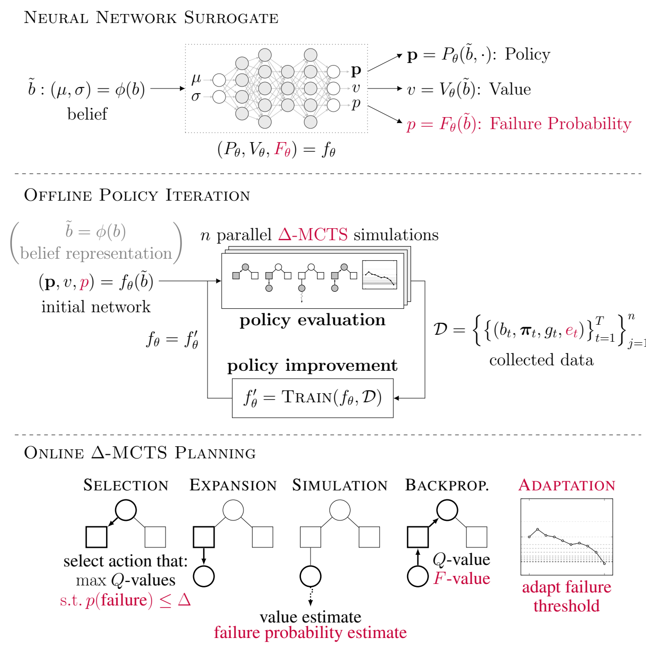


Figure 1: Elements of *ConstrainedZero* for CC-POMDP planning.

1 Introduction

When developing safety-critical agents to make sequential decisions in uncertain environments, planning and reinforcement learning algorithms often formulate the problem as a partially observable Markov decision process (POMDP) with the objective of maximizing a scalar-valued reward function [Kochenderfer *et al.*, 2022]. POMDP solution methods find a policies that maximizes this reward. To ensure adequate safety, the scalar reward is tuned to balance the goals of the agent while penalizing undesired behavior or failures. Recently, chance-constrained POMDPs (CC-POMDPs) have been used to frame the safe planning problem by separating the reward function into a constrained problem [Santana *et al.*, 2016]. The objective of CC-POMDPs is to maximize the goal rewards while satisfying the safety constraints.

To solve CC-POMDPs, online algorithms such as RAO* use heuristic forward search to find policies that maximize the reward and estimate the risk of constraint violation [Santana *et al.*, 2016]. RAO* plans over the reachable belief space for discrete state, action, and observation CC-POMDPs. The iterative RAO* (iRAO*) extends the heuristic search algorithm to multi-agent settings and handles continuous states and actions through Gaussian process regression and probabilistic flow tubes [Huang *et al.*, 2018]. Lauri *et al.* [2022] highlight the limitations of such chance-constrained POMDP algorithms and the need for scalable approaches to solve large-scale, long-horizon CC-POMDPs in practice.

To address scalability and applicability to continuous state and observation spaces, we introduce the *ConstrainedZero* policy iteration algorithm that combines offline neural network training of the value function, the action-selection policy, and the failure probability predictor with online Monte Carlo tree search (MCTS) to improve the policy through planning. ConstrainedZero is a direct extension to the POMDP belief-state planning algorithm BetaZero [Moss *et al.*, 2024] and the family of AlphaZero algorithms [Silver *et al.*, 2018], with extensions shown in red in fig. 1. Along with an open-source implementation,[1] our main contributions are threefold:

1. We introduce Δ-MCTS, an anytime algorithm for MDPs (applied to belief-state MDPs) that estimates failure probabilities along with Q-values and adjusts the failure probability threshold using adaptive conformal inference [Gibbs and Candes, 2021]. Δ-MCTS selects actions by maximizing the Q-value while satisfying that the failure probability constraint is below the adapted threshold using the introduced CC-PUCT criterion.
2. We introduce ConstrainedZero, a policy iteration algorithm that extends BetaZero for CC-POMDPs. ConstrainedZero includes an additional network head that estimates the failure probability given a belief and uses Δ-MCTS with the neural network surrogate to prioritize promising safe actions, replacing expensive rollouts or domain-specific heuristics. Framing the problem as a CC-POMDP means a target safety level can be specified instead of balancing penalties in the reward function.
3. We empirically evaluate ConstrainedZero and Δ-MCTS on three challenging safety-critical benchmark CC-POMDPs: a long-horizon localization task (LightDark [Platt Jr. *et al.*, 2010]), an aircraft collision avoidance system (modeled after ACAS X [Kochenderfer *et al.*, 2012]), and a CO_2 storage agent [Corso *et al.*, 2022].

2 Problem Formulation

This section formulates the safe planning problem as a belief-state CC-MDP. Background is also provided on Monte Carlo tree search and the BetaZero policy iteration algorithm.

POMDPs and belief-state MDPs. The partially observable Markov decision process (POMDP) is a framework for sequential decision making problems where the agent has uncertainty over their state in the environment [Kochenderfer *et al.*, 2022]. The POMDP is a 7-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, \gamma \rangle$ consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, an observation space $\mathcal{O}$, a transition model T, a reward model R, an observation model O, and a discount factor $\gamma \in [0, 1]$. When solving POMDPs, the objective is to find a policy $\pi(b)$ given a belief b over the unobserved state and return an action $a \in \mathcal{A}$ that maximizes the *value* of the belief, which is the expected discounted sum of rewards (i.e., the expected discounted *returns*) when continuing to follow the policy π:

$$\pi(b_0) = \operatorname*{arg\,max}_{a \in \mathcal{A}} \ \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_b(b_t, a_t) \mid b_0 \right] \tag{1}$$

where b_0 is the initial belief, often using the initial state distribution, and the belief-based reward is defined as

$$R_b(b, a) = \int_{s \in \mathcal{S}} b(s) R(s, a) \, \mathrm{d}s. \tag{2}$$

Every POMDP can be cast as an MDP by simply treating the belief as the state. In doing so, one can construct a belief-state MDP (BMDP) with the belief space $\mathcal{B}$ of the original POMDP as the MDP state space, while using the same action space $\mathcal{A}$, the belief-based reward model R_b from eq. (2), and a transition function $b' \sim T_b(\cdot \mid b, a)$ that takes the current belief b and action a and returns a stochastic updated belief b'. The belief transition function first samples a hidden state $s \sim b(\cdot)$ and transitions that state through the POMDP transition function $s' \sim T(\cdot \mid s, a)$. Then an observation is sampled from the observation model $o \sim O(\cdot \mid a, s')$ and finally the belief is updated to get the posterior

$$b'(s') \propto O(o \mid a, s') \int_{s \in \mathcal{S}} T(s' \mid s, a) b(s) \, \mathrm{d}s. \tag{3}$$

The belief update may be done exactly as in eq. (3) or using approximations such as a Kalman filter [Wan and Van Der Merwe, 2000] or particle filter [Thrun *et al.*, 2005].

The belief-state MDP tuple of $\langle \mathcal{B}, \mathcal{A}, T_b, R_b, \gamma \rangle$ can also be defined using a generative model $(b', r) \sim G_b(b, a)$ instead of an explicit belief transition model T_b and belief reward model R_b. The underlying POMDP can also use a generative model $(s', r, o) \sim G(s, a)$. Our work uses the generative POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, G, \gamma \rangle$ and the generative BMDP $\langle \mathcal{B}, \mathcal{A}, G_b, \gamma \rangle$.

Chance-constrained planning. When dealing with safety-critical sequential decision making problems, separating safety constraints from the objective allows for solvers to target an adequate level of safety while simultaneously maximizing rewards. This is in contrast to designing a single reward function to balance the rewards from the goals and penalties from violating safety. The chance-constrained POMDP (CC-POMDP) defines a failure set $\mathcal{F}$ that includes all state-action pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ that fail and a bound $\Delta \in [0, 1]$ on the probability, or chance, of a failure event occurring. Chance constraints are intuitive for users to define as they translate to the target failure probability of the agent, which is often the requirement for systems in industries such as aviation [Busch, 1985] and finance [Flannery, 1989]. The objective when solving CC-POMDPs is to maximize the value function while ensuring that the failure probability, or the chance constraint, is below the target threshold Δ:

$$\operatorname*{maximize}_{\pi} \ V^\pi(b_0) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_b(b_t, a_t) \mid b_0 \right] \tag{4}$$

$$\text{subject to} \ F^\pi(b_0) = \mathbb{P}_\pi \left[\bigvee_{t=0}^{\infty} \Big((s_t, a_t) \in \mathcal{F} \Big) \mid b_0 \right] \leq \Delta \tag{5}$$

The failure probability $F^\pi(b_t)$ is often called the *execution risk* of the policy π computed from the belief b_t.

Therefore, the CC-POMDP is defined as the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{F}, T, R, O, \gamma, \Delta \rangle$ which may also use a generative model G to replace $\langle T, R, O \rangle$. Our work casts the chance-constrained POMDP to a chance-constrained belief-MDP (CC-BMDP). The CC-BMDP tuple $\langle \mathcal{B}, \mathcal{A}, F_b, T_b, R_b, \gamma, \Delta \rangle$ extends BMDPs with an immediate failure probability function $F_b : \mathcal{B} \times \mathcal{A} \to [0, 1]$ and a failure probability threshold Δ. The immediate failure probability is computed as

$$F_b(b, a) = \int_{s \in \mathcal{S}} b(s) \mathbb{1}\big\{(s, a) \in \mathcal{F}\big\} \, \mathrm{d}s \tag{6}$$

using the failure set $\mathcal{F}$. The CC-BMDP can also be defined with a generative model $(b', r, p) \sim G_b(b, a)$ that also returns the failure probability $p = F_b(b, a)$ with notation overloading of G_b, resulting in the tuple $\langle \mathcal{B}, \mathcal{A}, G_b, \gamma, \Delta \rangle$.

[1] https://github.com/sisl/ConstrainedZero.jl

Monte Carlo tree search. The best-first search algorithm, Monte Carlo tree search (MCTS), is designed to solve MDPs [Coulom, 2007] and has been applied to solve POMDPs cast as belief-state MDPs [Sunberg and Kochenderfer, 2018; Fischer and Tas, 2020; Moss *et al.*, 2024]. MCTS is an online algorithm that determines the best action to take from the current state s (or belief state b). Starting from the root state, MCTS iteratively simulates the following four steps to build out a tree of possible reachable futures to a depth d:

1. **Selection.** An action is selected from the existing children of the current state node or sampled from the action space. The selection process balances exploration and exploitation. Metrics such as UCT [Kocsis and Szepesvári, 2006] or PUCT [Silver *et al.*, 2017] have been used in the literature to select the action to take.
2. **Expansion.** Once an action is selected, it is executed from the current state node to expand the tree. For stochastic state transitions, methods like progressive widening [Couëtoux *et al.*, 2011] or state abstraction refinement [Sokota *et al.*, 2021] can be used to control when to execute the action or take an existing tree path.
3. **Simulation.** From the expanded state, the tree is recursively built from this new root node. Simulation returns an estimate of the value of the expanded state. The value estimate could use a rollout policy [Silver *et al.*, 2016] (which may be expensive for BMDPs), or use function approximators such as neural networks [Silver *et al.*, 2017; Fischer *et al.*, 2022; Moss *et al.*, 2024].
4. **Backpropagation.** Finally, the value estimate is combined with the immediate reward to get the Q-value. This Q-value is assigned to the parent state-action node as a running mean. This process backpropagates the signal up the tree path that led to that node.

After a prescribed number of iterations, or at anytime as determined by a compute-time constraint, MCTS will select the best action from the children of the root node, often using the Q-values or visit counts [Browne *et al.*, 2012].

Brázdil *et al.* [2020] introduce an algorithm for CC-MDPs that uses UCT with a table-based value and risk predictor, and a linear program to compute an action distribution that satisfies an adaptive constraint. Ayton and Williams [2018] introduce an MCTS algorithm for CC-MDPs that selects actions that satisfy a local chance constraint over state histories and prune branches that violate the constraint.

MCTS algorithms have also been developed for cost-constrained POMDPs (C-POMDPs) [Isom *et al.*, 2008], where a discounted cost is minimized. Algorithms such as C-POMCP [Lee *et al.*, 2018], C-MCTS [Parthasarathy *et al.*, 2023], C-POMCPOW, C-PTF-DPW, and C-POMCP-DPW [Jamgochian *et al.*, 2023] estimate the cost function during search using rollouts, while our approach uses *chance* constraints and estimates constraint violation *probabilities* using neural network surrogates.

BetaZero. The BetaZero belief-state planning algorithm extends AlphaZero to long-horizon POMDPs [Moss *et al.*, 2024]. BetaZero combines MCTS planning in belief space with neural network surrogates of the value function and action-selection policy. The neural network takes as input the belief, which is converted to an input representation $\tilde{b} = \phi(b) = (\mu, \sigma)$ of summary statistics, and estimates the value $v = V_\theta(\tilde{b})$ and policy vector $\mathbf{p} = P_\theta(\tilde{b})$ as two heads of the network $(\mathbf{p}, v) = f_\theta(\tilde{b})$. BetaZero improves the neural network through policy iteration by iterating the following:

1. **Policy evaluation**: Execute parallel MCTS episodes using the current network to collect training data for policy imitation learning and value regression.
2. **Policy improvement**: Use the recent MCTS data over a specified window to re-train the neural network.

Learning from experience through policy iteration reduces the required MCTS depth of search (by estimating the value of future belief states) and breadth of search (by selecting actions prioritized by the policy head of the network). The neural network surrogate also acts as a learned replacement for domain-specific heuristics of the value function and policy.

In BetaZero, the root node action selection used during MCTS, which is also the trained policy vector target, is a combination of the observed Q-values and visit counts N seen during search. This combination incorporates all available information in the tree, given that belief-state planning is often limited due to the expensive belief updates that occur every state transition. The root node action is selected according to the Q-weighted policy vector:

$$\boldsymbol{\pi}_{\text{tree}}(b, a) \propto \left(\left(\frac{\exp Q(b,a)}{\sum_{a'} \exp Q(b,a')} \right) \left(\frac{N(b,a)}{\sum_{a'} N(b,a')} \right) \right)^{1/\tau} \tag{7}$$

where τ controls the sampling temperature (used during policy iteration) and returns the argmax when $\tau \to 0$ (used during final evaluation). Intuitively, the Q-weighted policy vector uses information from what was *found* during search (Q-values) and what the search *focused on* (visit counts). Moss *et al.* [2024] show that the combination leads to the highest return on various benchmark POMDPs.

3 Approach

ConstrainedZero follows the BetaZero policy iteration steps of *policy evaluation* and *policy improvement* while also collecting failure event indicators to train the failure probability network head, shown in red in algorithm 1. During policy evaluation, n parallel Δ-MCTS executions are run and a data set $\mathcal{D}$ is collected. The data set $\mathcal{D} = \left\{ \{b_t, \boldsymbol{\pi}_t, g_t, e_t\}_{t=1}^T \right\}_{j=1}^n$ is a tuple of the belief at episode time step t denoted b_t, the tree policy $\boldsymbol{\pi}_t$, the return $g_t = \sum_{i=t}^T \gamma^{(i-t)} r_i$ based on the observed reward r_i and discount factor γ, and the failure event indicator e_t, where g_t and e_t are computed at the end of the trajectory for all time $t \le T$. The failure event is computed as the disjunction of all state and action pairs of the CC-POMDP in the execution trajectory to ensure that if a trajectory failed at some point the full trajectory is marked as a failure:

$$e_t = \mathbb{1}\left\{ \bigvee_{i=t}^{T} \Big((s_i, a_i) \in \mathcal{F} \Big) \right\} \tag{8}$$

where $\mathbb{1}\{E\}$ is the indicator function that returns 1 when event E is true and 0 otherwise.

Algorithm 1 Offline ConstrainedZero policy iteration.

Require: $\mathcal{P}_{\text{cc}} \stackrel{\text{def}}{=} \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{F}, G, \gamma, \Delta_0 \rangle$ ▷ Generative CC-POMDP
Require: ψ: Parameters
function POLICYITERATION($\mathcal{P}_{\text{cc}}, \psi$)
  $f_\theta \leftarrow$ INITIALIZENETWORK(ψ) ▷ $(P_\theta, V_\theta, F_\theta) \leftarrow f_\theta$
  for $i \leftarrow 1$ **to** $n_{\text{iterations}}$
    $\mathcal{D} \leftarrow$ COLLECTDATA($\mathcal{P}_{\text{cc}}, f_\theta$) ▷ policy evaluation
    $f_\theta \leftarrow$ TRAIN($f_\theta, \mathcal{D}$) ▷ policy improvement
  return $\pi(b) \leftarrow \Delta$MCTS($\mathcal{P}_{\text{cc}}, f_\theta, b$) ▷ online policy (alg. 2)
function COLLECTDATA($\mathcal{P}_{\text{cc}}, f_\theta$)
  $\mathcal{D} = \emptyset$
  parallel for $j \leftarrow 1$ **to** n_{data}
    for $t \leftarrow 1$ **to** T ▷ sample a trajectory
      $a_t \leftarrow \Delta$MCTS($\mathcal{P}_{\text{cc}}, f_\theta, b_t$) ▷ record $\boldsymbol{\pi}_{\text{tree}}^{(t)}$ vector
      $s_{t+1}, r_t, o_t \sim G(s_t, a_t)$ ▷ generative model
      $b_{t+1} \leftarrow P(s_{t+1} \mid b_t, a_t, o_t)$ ▷ update belief
    for $t \leftarrow 1$ **to** T
      $g_t \leftarrow \sum_{i=t}^{T} \gamma^{(i-t)} r_i$ ▷ computed over trajectory
      $e_t \leftarrow \mathbb{1}\left\{ \bigvee_{i=t}^{T} \left((s_i, a_i) \in \mathcal{F}\right) \right\}$ ▷ failure event
      $\mathcal{D}_j^{(t)} \leftarrow \mathcal{D}_j^{(t)} \cup \left\{ (b_t, \boldsymbol{\pi}_{\text{tree}}^{(t)}, g_t, e_t) \right\}$
  return $\mathcal{D}$

During policy improvement, the neural network is trained to minimize the MSE or MAE loss $\mathcal{L}_{V_\theta}(g_t, v_t)$ to regress the value function $v_t = V_\theta(\tilde{b}_t)$, minimize the cross-entropy loss $\mathcal{L}_{P_\theta}(\boldsymbol{\pi}_t, \mathbf{p}_t)$ to imitate the tree policy $\mathbf{p}_t = P_\theta(\tilde{b}_t)$, and additionally minimize the binary cross-entropy loss $\mathcal{L}_{F_\theta}(e_t, p_t)$ to regress the failure probability function $p_t = F_\theta(\tilde{b}_t)$, with added regularization using the L_2-norm of the weights θ:

$$\mathcal{L}_{V_\theta}(g_t, v_t) = (g_t - v_t)^2 \text{ or } |g_t - v_t|$$
$$\mathcal{L}_{P_\theta}(\boldsymbol{\pi}_t, \mathbf{p}_t) = -\boldsymbol{\pi}_t^\top \log \mathbf{p}_t$$
$$\mathcal{L}_{F_\theta}(e_t, p_t) = -e_t \log p_t - (1 - e_t) \log(1 - p_t)$$
$$\ell_{\text{CZ}} = \mathcal{L}_{V_\theta}(g_t, v_t) + \mathcal{L}_{P_\theta}(\boldsymbol{\pi}_t, \mathbf{p}_t) + \mathcal{L}_{F_\theta}(e_t, p_t) + \lambda\|\theta\|^2$$

The failure probability head of the neural network includes a final sigmoid layer to ensure the output can be interpreted as a probability in the range $[0, 1]$.

3.1 Adaptive Safety Constraints in Δ-MCTS

When using online MCTS for CC-BMDP planning, two considerations have to be addressed: 1) how to estimate the observed failure probability in the tree search, and 2) how to select actions constrained by this failure probability.

At each node for the belief-state and action (b, a), the immediate failure probability p is computed using the generative model (or by calling $p = F_b(b, a)$ directly). An estimate of the future failure probability p' can be computed using rollouts, which may be expensive for belief-state planning, thus we use the trained neural network head for failure probability estimation $p' = F_\theta(\tilde{b}')$. Similar to the Q-value, we must compute the full failure probability of the trajectory from the immediate time step to the horizon, termed the F-value. Let E be the immediate failure event from belief b when taking action a at time t, and let E' be the event of failing in the future (from $t + 1$ to the horizon T). The probability of failing between the current time t and the horizon T becomes:

$$\begin{aligned} P(E_{t:T}) &= P(E) + P(E') - P(E \cap E') & (9)\\ &= P(E) + P(E') - P(E')P(E \mid E') & (10)\\ &= P(E) + P(E') - P(E')P(E) & (11)\\ &= p + p' - pp' & (12)\\ &= p + (1-p)p' & (13)\end{aligned}$$

assuming independence in eq. (11). A discount δ is applied to control the influence of the future failure probability:

$$p = p + \delta(1 - p)p' \tag{14}$$

Unlike Carpin and Thayer [2022], who backup F-values based on the best-case, we backpropagate the F-values up the tree similar to Q-values (alg. 2, line 16):

$$F(b, a) = F(b, a) + \frac{p - F(b, a)}{N(b, a)} \tag{15}$$

which is a running mean estimate where $F(b, a)$ is initialized using the initialization function $F_0(b, a)$ (noting the F_0 subscript: which could either be zero, the immediate failure probability $F_b(b, a)$, or the bootstrapped value by taking action a to get a new belief b' and computing eq. (14) based on the $p' = F_\theta(\tilde{b}')$ estimate). Note, the number of times the node (b, a) is visited is indicated as the visit count $N(b, a)$.

Using the estimate $F(b, a)$, a simple way to select actions that do not violate the safety constraint set by Δ would be to use the PUCT algorithm [Silver *et al.*, 2018]

$$\pi_{\text{explore}}(b) = \operatorname*{arg\,max}_{a \in A(b)} \bar{Q}(b, a) + c\left(P_\theta(\tilde{b}, a)\frac{\sqrt{N(b)}}{1 + N(b, a)}\right) \tag{16}$$

with a hard constraint on safety of only choosing actions such that $F(b, a) \leq \Delta$ is satisfied. PUCT exploits nodes based on their observed Q-values and explores nodes based on their visit counts weighted by the action-selection policy P_θ to explore promising actions.[2]

However, if the failure probability threshold Δ is too conservative, the action-selection process may fail to find *any* action that satisfies the constraint. Therefore, Δ-MCTS tracks an estimate of the threshold $\Delta(b)$ for each belief node and updates it using *adaptive conformal inference* (ACI) [Gibbs and Candes, 2021]. ACI is a statistical method that provides valid prediction intervals without assumptions on how the time-series data was generated. The adaptive threshold is initialized to the target tolerance $\Delta(b) = \Delta_0$ where $\Delta_0 = \Delta$ from the CC-BMDP. Each time the F-value is updated (either by eq. (15) or initialization), the ADAPTATION procedure is called to update the current acceptable safety threshold.

[2]The belief node visit count is $N(b) = \sum_{a'} N(b, a')$ for children $a' \in A(b)$. Following Schrittwieser *et al.* [2020], we normalize the Q-values between zero and one, denoted $\bar{Q}$, to avoid problem-specific heuristics when selecting an exploration constant c:

$$\bar{Q}(b, a) = \frac{Q(b, a) - \min_{(b', a') \in \mathcal{T}} Q(b', a')}{\max_{(b', a') \in \mathcal{T}} Q(b', a') - \min_{(b', a') \in \mathcal{T}} Q(b', a')} \tag{17}$$

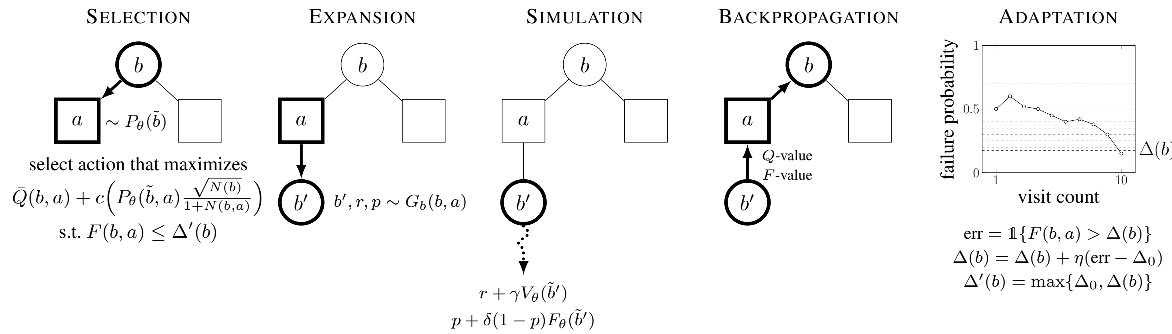


Figure 2: *ConstrainedZero* online Monte Carlo tree search with failure threshold adaptation (Δ-MCTS).

In adaptation, the error term of $\text{err} = \mathbb{1}\{F(b,a) > \Delta(b)\}$ indicates when to widen or restrict the estimated threshold $\Delta(b)$ based on whether the failure probability estimate of the most recently explored belief-action node $F(b,a)$ is above or below the current threshold. The estimated threshold is updated according to

$$\Delta(b) = \Delta(b) + \eta(\text{err} - \Delta_0) \tag{18}$$

which will widen the threshold if the observed failure probability is outside the threshold (i.e., if the error is one), and will tighten the threshold otherwise:

$$\Delta(b) = \begin{cases} \Delta(b) + \eta(1-\Delta_0) & \text{if } F(b,a) > \Delta(b) \\ \Delta(b) - \eta\Delta_0 & \text{if } F(b,a) \leq \Delta(b) \end{cases} \tag{19}$$

Intuitively, the update adjusts the threshold of acceptable failure probability $\Delta(b)$ based on recent experience. If the failure probability $F(b,a)$ for a recent action is higher than the current threshold $\Delta(b)$, this indicates a higher risk than expected. Thus, the threshold is increased by $\eta(1-\Delta_0)$ for $\eta > 0$ to allow for more risk in future actions. Otherwise, if $F(b,a)$ is lower than the threshold, this means actions are safer than expected and the threshold is decreased by $\eta\Delta_0$ (favoring a more reactive increase than decrease of the threshold). Notably, Gibbs and Candes [2021] prove that $\Delta(b)$ converges exactly to the desired target over time.

We clip the final threshold to the lower and upper bounds of the observed failure probability for a given belief b to restrict

Algorithm 2 ConstrainedZero online Δ-MCTS.

function ΔMCTS($\mathcal{P}_{\text{cc}}, f_\theta, b$)
  $\mathcal{M}_{\text{cc}} \leftarrow \langle \mathcal{B}, \mathcal{A}, G_b, \gamma, \Delta_0 \rangle$ ▷ convert from generative $\mathcal{P}_{cc}$
  for 1 **to** n_{online}
    SIMULATE(f_θ, b, d)
  $\boldsymbol{\pi}_{\text{tree}}(b,a) \propto \left(\left(\frac{\exp Q(b,a)}{\sum_{a'} \exp Q(b,a')}\right)\left(\frac{N(b,a)}{\sum_{a'} N(b,a')}\right)\right)^{1/\tau}$
  return $a \sim \boldsymbol{\pi}_{\text{tree}}(b,a)$ ▷ let $\tau \to 0$ to get argmax
    s.t. $F(b,a) \leq \Delta'(b)$ ▷ $\Delta'(b) = \max\{\Delta_0, \Delta(b)\}$

function SIMULATE(f_θ, b, d)
  if TERMINAL(b) **return** $0, 0$
  if $b \notin \mathcal{T}$ **or** $d = 0$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{b\}$
    $N(b) \leftarrow N_0(b)$
    $\tilde{b} \leftarrow \phi(b)$ ▷ input belief representation
    return $V_\theta(\tilde{b}), F_\theta(\tilde{b})$ ▷ network lookup
  $N(b) \leftarrow N(b) + 1$
  $a \leftarrow$ ACTIONSELECTION(f_θ, b)
  $(b', r, p) \leftarrow$ BELIEFSTATEEXPANSION(b, a)
  $v', p' \leftarrow$ SIMULATE($f_\theta, b', d-1$)
  $q \leftarrow r + \gamma v'$
  $p \leftarrow p + \delta(1-p)p'$ ▷ observed failure probability
  $N(b,a) \leftarrow N(b,a) + 1$
  $Q(b,a) \leftarrow Q(b,a) + \frac{q - Q(b,a)}{N(b,a)}$ ▷ backpropagate Q-value
  $F(b,a) \leftarrow F(b,a) + \frac{p - F(b,a)}{N(b,a)}$ ▷ backpropagate F-value
  ADAPTATION(Δ, b, a)
  return q, p

Algorithm 3 Δ-MCTS selection, expansion, and adaptation.

function ACTIONSELECTION(f_θ, b)
  $\tilde{b} \leftarrow \phi(b)$
  if $|A(b)| \leq k_a N(b)^{\alpha_a}$ ▷ action progressive widening
    $a \sim P_\theta(\tilde{b}, \cdot)$ ▷ prioritized from network
    if $a \notin A(b)$
      $N(b,a) \leftarrow N_0(b,a)$
      $Q(b,a) \leftarrow Q_0(b,a)$
      $F(b,a) \leftarrow F_0(b,a)$
      $\Delta(b) \leftarrow \Delta_0$ ▷ target tolerance Δ_0
      ADAPTATION(Δ, b, a)
      $A(b) \leftarrow A(b) \cup \{a\}$
  return $\underset{a \in A(b)}{\arg\max}\ \bar{Q}(b,a) + c\Big(P_\theta(\tilde{b},a)\frac{\sqrt{N(b)}}{1+N(b,a)}\Big)$
    s.t. $F(b,a) \leq \Delta'(b)$ ▷ CC-PUCT

function BELIEFSTATEEXPANSION(b, a)
  if $|B(b,a)| \leq k_b N(b,a)^{\alpha_b}$ ▷ belief progressive widening
    $b', r, p \sim G_b(b,a)$ ▷ generative model
    $B(b,a) \leftarrow B(b,a) \cup \{(b', r, p)\}$
  else
    $b', r, p \sim B(b,a)$
  return b', r, p

function ADAPTATION(Δ, b, a)
  $l(b) \leftarrow \min_{a' \in A(b)} F(b,a')$ ▷ update bounds
  $u(b) \leftarrow \max_{a' \in A(b)} F(b,a')$
  $\text{err} \leftarrow \mathbb{1}\{F(b,a) > \Delta(b)\}$
  $\Delta(b) \leftarrow \text{clip}\big(\Delta(b) + \eta(\text{err} - \Delta_0), l(b), u(b)\big)$

the change in $\Delta(b)$ and, more importantly, to guarantee that at least one action is available for selection:

$$\Delta(b) = \text{clip}\big(\Delta(b) + \eta(\text{err} - \Delta_0),\ l(b),\ u(b)\big) \tag{20}$$

with the lower and upper bounds of $l(b) = \min_{a'} F(b,a')$ and $u(b) = \max_{a'} F(b,a')$ for a' in children nodes $A(b)$.

The resulting criterion selects actions that satisfy the adaptive constraint of $F(b,a) \le \Delta'(b)$ where the selection threshold $\Delta'(b) = \max\{\Delta_0, \Delta(b)\}$ upper bounds the failure probability. Together, the Δ-MCTS exploration policy becomes:

$$\pi_{\text{explore}}(b) = \operatorname*{arg\,max}_{a \in A(b)} \bar{Q}(b,a) + c\Big(P_\theta(\tilde{b},a)\frac{\sqrt{N(b)}}{1+N(b,a)}\Big) \tag{21}$$

$$\text{s.t. } F(b,a) \le \Delta'(b) \tag{22}$$

termed the *chance-constrained PUCT* criterion (CC-PUCT). The constraint in eq. (22) is also used in root node action selection (line 7, alg. 2). In practice, π_{explore} is computed using the indicator $\mathbb{1}$, returning the action $a \in A(b)$ that maximizes:

$$\mathbb{1}\big\{F(b,a) \le \Delta'(b)\big\}\Big(\bar{Q}(b,a) + c\Big(P_\theta(\tilde{b},a)\frac{\sqrt{N(b)}}{1+N(b,a)}\Big)\Big)$$

The benefit of CC-PUCT is that when our explored samples satisfy the constraint $\Delta'(b)$ (defined over the belief rather than both belief and action) we may explore new actions from this belief which are both safe and have the potential for higher reward. The key idea is that actions are chosen based on the balance between safety and utility; ensuring that we do not over-prioritize safety at the expense of potential rewards, while not exploiting rewards without regarding the risk.

The five stages of Δ-MCTS are shown in algorithms 2–3: *selection*, *expansion*, *simulation*, *backpropagation*, and *adaptation*, with extensions to BetaZero shown in red.

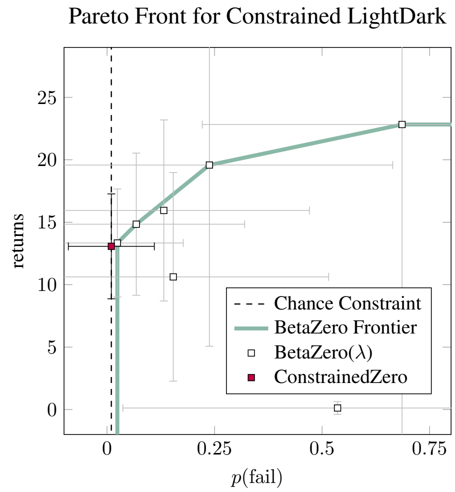


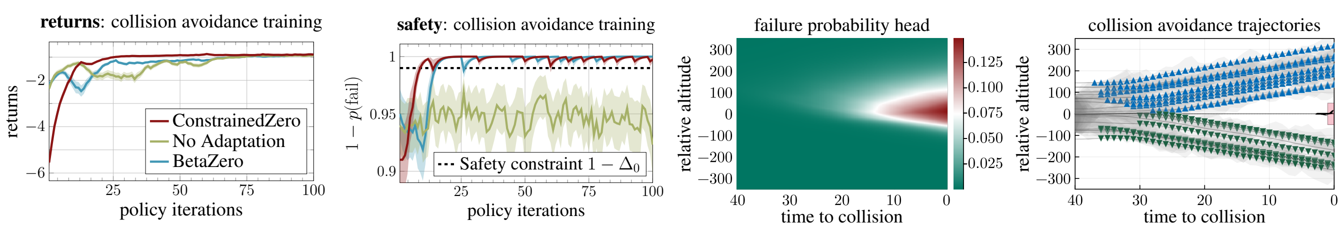
Figure 3: BetaZero(λ) comparison.

4 Experiments

For a fair comparison, ConstrainedZero was evaluated against BetaZero using the same network and MCTS parameters. BetaZero uses a scalarized reward function to penalize failures, while ConstrainedZero omits the penalty and plans using the adaptive safety constraint instead. The BetaZero reward takes the form $\bar{R}_b(b,a) = R_b(b,a) - \lambda C(b,a)$ with a cost C scaled by λ. Three safety-critical CC-POMDPs were evaluated. The first is the *LightDark* POMDP, a standard benchmark localization task [Platt Jr. *et al.*, 2010], where a particle filter is used to update the belief with $n_{\text{particles}} = 500$. A failure occurs if the agent stops outside of the goal (i.e., the origin). The next CC-POMDP is the *aircraft collision avoidance* problem (CAS), modeled after ACAS X [Kochenderfer *et al.*, 2012]. In the CAS problem, the ownship aircraft attempts to avoid a near mid-air collision (NMAC) with an intruding aircraft while minimizing the alert and reversal rates. An NMAC occurs if the ownship is within 50 meters in relative altitude h_{rel} of the intruder at the end of the encounter (i.e., when $|h_{\text{rel}}| \le 50$ and time to collision $\tau = 0$). The belief is updated with an unscented Kalman filter to track the mean and covariance of the state variables [Wan and Van Der Merwe, 2000]. Lastly, we study safe *carbon capture and storage* (CCS), which is a promising mitigation of global emissions that captures CO_2 and stores it in porous subsurface material [Corso *et al.*, 2022]. A challenge of CCS is safely injecting CO_2 into the subsurface while mitigating risk of leakage and earthquakes, where any CO_2 leakage indicates a failure. The simplified CCS problem uses spillpoint analysis to model the top surface of the injection site and a sequential importance resampling particle filter is updated with observations at drilling locations [Corso *et al.*, 2022].

4.1 Empirical Results

Figure 3 compares ConstrainedZero against BetaZero, where BetaZero uses different values of the penalty λ. The penalties were swept between -10 and -1000 with -100 being the standard for the LightDark POMDP (proportional to the goal reward of 100). A target safety level of $\Delta_0 = 0.01$ was chosen for ConstrainedZero. ConstrainedZero exceeds the



(a) Returns from policy iteration. (b) Safety from policy iteration. (c) Higher p(fail) near an NMAC. (d) Climb (blue), descend (green).

Figure 4: Results for the collision avoidance CC-POMDP. Figure 4d matches the "notch" behavior from Kochenderfer *et al.* [2012].

	LightDark $\Delta_0 = 0.01$		Collision Avoidance $\Delta_0 = 0.01$		Spillpoint CCS $\Delta_0 = 0.05$	
	$p(\text{fail}) \downarrow$	returns $\uparrow$	$p(\text{fail}) \downarrow$	returns $\uparrow$	$p(\text{fail}) \downarrow$	returns $\uparrow$
ConstrainedZero	$\mathbf{0.01}_{\pm 0.01}$	$\mathbf{13.07}_{\pm 0.42}$	$\mathbf{0.00}_{\pm 0.00}$	$\mathbf{-0.74}_{\pm 0.03}$	$\mathbf{0.05}_{\pm 0.02}$	$\mathbf{2.62}_{\pm 0.12}$
No Adaptation*	$0.66_{\pm 0.05}$	$27.47_{\pm 3.90}$	$0.03_{\pm 0.02}$	$-1.00_{\pm 0.00}$	$0.69_{\pm 0.04}$	$6.18_{\pm 0.36}$
Δ-MCTS (no f_θ)†	$0.01_{\pm 0.01}$	$1.86_{\pm 0.20}$	$0.32_{\pm 0.05}$	$0.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$6.87_{\pm 0.50}$
Raw Policy P_θ	$0.01_{\pm 0.01}$	$12.88_{\pm 0.46}$	$0.00_{\pm 0.00}$	$-0.86_{\pm 0.02}$	$0.06_{\pm 0.02}$	$2.45_{\pm 0.11}$
Raw Value‡ V_θ	$0.72_{\pm 0.05}$	$28.00_{\pm 4.51}$	$0.16_{\pm 0.04}$	$-0.20_{\pm 0.04}$	$0.38_{\pm 0.05}$	$4.27_{\pm 0.30}$
Raw Failure‡ F_θ	$0.80_{\pm 0.04}$	$0.05_{\pm 0.04}$	$0.00_{\pm 0.00}$	$-1.62_{\pm 0.08}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$

All results report the mean $\pm$ standard error over 100 seeds, evaluated using the argmax of eq. (7), i.e., $\tau \to 0$.
* Trained with the same parameters as ConstrainedZero without adaptation, i.e., only a hard constraint on Δ_0.
† Δ-MCTS without the neural network for the value or failure probability and a random policy for CC-PUCT.
‡ One-step look-ahead over all actions using only the value or failure probability network head with 5 obs. per action.

Table 1: ConstrainedZero results. Bold indicates the best results within the Δ_0 threshold.

BetaZero Pareto curve and achieves the target level of safety with a failure probability of 0.01 ± 0.01 computed over 100 episodes. Notably, ConstrainedZero also has less variance in the failure probability and the returns. BetaZero still achieves good performance but at the cost of sweeping the penalty values without explicitly defining a safety threshold to satisfy.

Shown in table 1, an ablation study is conducted for ConstrainedZero. Most notably, the adaptation procedure is crucial to enable the algorithm to properly balance safety and utility during planning (also shown in fig. 4a–4b). When comparing Δ-MCTS without network approximators against ConstrainedZero, it is clear that offline policy iteration allows for better online planning. Using only the raw policy head P_θ achieves good performance, which is trained to imitate the tree policy. However, incorporating additional online planning with the full network yields better results overall, enabling planning over potentially unseen information. The full ConstrainedZero algorithm consistently achieves the highest return within the satisfied target level of safety Δ_0.

Compared to BetaZero, fig. 4a and fig. 4b highlight that ConstrainedZero satisfies the safety constraint earlier during policy iteration, while simultaneously maximizing returns (shown for the CAS problem). The policy trained without adaptation learns to maximize returns but fails to satisfy the safety constraint. This is because without adaptation, the algorithm will attempt to satisfy a fixed constraint, not taking into account the outcomes of its actions. With adaptation, ConstrainedZero adjusts the constraint in response to feedback from the environment, resulting in the algorithm becoming more capable at optimizing its performance within the bounds of the adaptive constraint. This demonstrates the importance of adaptation, as a fixed constraint may be too conservative or too risky, leading to suboptimal decision-making.

5 Conclusions

This work introduces *ConstrainedZero*, an extension of the BetaZero POMDP planning algorithm to chance-constrained POMDPs. Along with neural network estimates of the value function and action-selection policy, we include a network head that estimates the failure probability given a belief. By formulating the safe planning problem as a CC-POMDP, we select a target level of safety to optimize towards, instead of traditional POMDP methods that tune the reward function to balance safety and utility as a multi-objective problem. We develop an extension to Monte Carlo tree search that includes an *adaptation* stage that adjusts the target level of safety during planning with adaptive conformal inference. The resulting Δ-MCTS algorithm modifies MCTS for CC-POMDPs and addresses the issue of overfitting to failure predictions. Additionally, a constrained action-selection criterion (CC-PUCT) was developed to enable planning under constraints. The full implementation and experiments are open sourced and are part of the Julia package BetaZero.jl.[3]

Limitations. Adapting the safety level when planning with approximations may lead to deviations despite ACI convergence guarantees, but the lack of adaptation also does not guarantee the desired safety. However, our experiments show that this flexibility helps the algorithm find policies matching the targeted safety. Similar to BetaZero, ConstrainedZero may require more computing resources than existing POMDP solvers due to neural network training and parallel Δ-MCTS episodes. However, it is designed for large-scale, uncertain problems in high-dimensional spaces that require long-horizon planning. We focus on real-world scenarios where transition dynamics, not policy training, are the main challenge, using past experiences to learn an approximately optimal policy offline that is refined online using tree search.

Future work. A benefit of Δ-MCTS unexplored in this work is that it also applies directly to safety-critical MDPs, such as scheduling [Soualhia *et al.*, 2020]. Future work could focus on the application of ConstrainedZero to fully observable MDP settings and extending the algorithm to work over multiple failure modes. Other future work could improve the data efficiency of ConstrainedZero, extend it to continuous actions similar to algorithms such as A0C [Moerland *et al.*, 2018], or apply it to safety-critical robotics tasks such as navigation [Ong *et al.*, 2010] and object manipulation [Pajarinen and Kyrki, 2017; Pajarinen *et al.*, 2022].

[3]https://github.com/sisl/ConstrainedZero.jl

Acknowledgments

This research is funded by OMV and J.F. thanks the Karlsruhe House of Young Scientists (KHYS) for travel grant funding.

References

[Ayton and Williams, 2018] Benjamin J. Ayton and Brian C. Williams. Vulcan: A Monte Carlo Algorithm for Large Chance Constrained MDPs with Risk Bounding Functions. *arXiv:1809.01220*, 2018.

[Brázdil *et al.*, 2020] Tomáš Brázdil, Krishnendu Chatterjee, Petr Novotnỳ, and Jiří Vahala. Reinforcement Learning of Risk-Constrained Policies in Markov Decision Processes. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 9794–9801, 2020.

[Browne *et al.*, 2012] Cameron B. Browne, Edward Powley, et al. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[Busch, 1985] Allen C. Busch. *Methodology for Establishing A Target Level of Safety*. Federal Aviation Administraton Technical Center, 1985.

[Carpin and Thayer, 2022] Stefano Carpin and Thomas C. Thayer. Solving Stochastic Orienteering Problems with Chance Constraints Using Monte Carlo Tree Search. In *International Conference on Automation Science and Engineering (CASE)*, pages 1170–1177. IEEE, 2022.

[Corso *et al.*, 2022] Anthony Corso, Yizheng Wang, Markus Zechner, Jef Caers, and Mykel J. Kochenderfer. A POMDP Model for Safe Geological Carbon Sequestration. *NeurIPS Workshop on Tackling Climate Change with Machine Learning*, 2022.

[Couëtoux *et al.*, 2011] Adrien Couëtoux, Jean-Baptiste Hoock, et al. Continuous Upper Confidence Trees. In *Learning and Intelligent Optimization*. Springer, 2011.

[Coulom, 2007] Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Computers and Games*, 2007.

[Fischer and Tas, 2020] Johannes Fischer and Ömer Sahin Tas. Information Particle Filter Tree: An Online Algorithm for POMDPs with Belief-Based Rewards on Continuous Domains. In *International Conference on Machine Learning*, pages 3177–3187. PMLR, 2020.

[Fischer *et al.*, 2022] Johannes Fischer, Etienne Bührle, Danial Kamran, and Christoph Stiller. Guiding Belief Space Planning with Learned Models for Interactive Merging. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 2542–2549, 2022.

[Flannery, 1989] Mark J. Flannery. Capital Regulation and Insured Banks Choice of Individual Loan Default Risks. *Journal of Monetary Economics*, 24(2):235–258, 1989.

[Gibbs and Candes, 2021] Isaac Gibbs and Emmanuel Candes. Adaptive Conformal Inference Under Distribution Shift. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:1660–1672, 2021.

[Huang *et al.*, 2018] Xin Huang, Ashkan Jasour, et al. Hybrid Risk-Aware Conditional Planning with Applications in Autonomous Vehicles. In *IEEE Conference on Decision and Control (CDC)*, pages 3608–3614. IEEE, 2018.

[Isom *et al.*, 2008] Joshua D. Isom, Sean P. Meyn, and Richard D. Braatz. Piecewise Linear Dynamic Programming for Constrained POMDPs. In *AAAI Conference on Artificial Intelligence*, volume 1, pages 291–296, 2008.

[Jamgochian *et al.*, 2023] Arec Jamgochian, Anthony Corso, and Mykel J. Kochenderfer. Online Planning for Constrained POMDPs with Continuous Spaces through Dual Ascent. *International Conference on Automated Planning and Scheduling (ICAPS)*, 33(1):198–202, 2023.

[Kochenderfer *et al.*, 2012] Mykel J. Kochenderfer, Jessica E. Holland, and James P. Chryssanthacopoulos. Next-Generation Airborne Collision Avoidance System. *Lincoln Laboratory Journal*, 19(1), 2012.

[Kochenderfer *et al.*, 2022] Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray. *Algorithms for Decision Making*. MIT Press, 2022.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *European Conference on Machine Learning*. Springer, 2006.

[Lauri *et al.*, 2022] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially Observable Markov Decision Processes in Robotics: A Survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[Lee *et al.*, 2018] Jongmin Lee, Geon-Hyeong Kim, Pascal Poupart, and Kee-Eung Kim. Monte-Carlo Tree Search for Constrained POMDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[Moerland *et al.*, 2018] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. A0C: Alpha Zero in Continuous Action Space. In *ICML Planning and Learning (PAL) Workshop*, 2018.

[Moss *et al.*, 2024] Robert J. Moss, Anthony Corso, Jef Caers, and Mykel J. Kochenderfer. BetaZero: Belief-State Planning for Long-Horizon POMDPs using Learned Approximations. *Reinforcement Learning Journal (RLJ)*, 1(1), 2024.

[Ong *et al.*, 2010] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research*, 29(8), 2010.

[Pajarinen and Kyrki, 2017] Joni Pajarinen and Ville Kyrki. Robotic Manipulation of Multiple Objects as a POMDP. *Artificial Intelligence*, 247:213–228, 2017.

[Pajarinen *et al.*, 2022] Joni Pajarinen, Jens Lundell, and Ville Kyrki. POMDP Planning Under Object Composition Uncertainty: Application to Robotic Manipulation. *IEEE Transactions on Robotics*, 39(1):41–56, 2022.

[Parthasarathy *et al.*, 2023] Dinesh Parthasarathy, Georgios Kontes, Axel Plinge, and Christopher Mutschler. C-MCTS: Safe Planning with Monte Carlo Tree Search. *arXiv:2305.16209*, 2023.

[Platt Jr. *et al.*, 2010] Robert Platt Jr., Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief Space Planning Assuming Maximum Likelihood Observations. *Robotics: Science and Systems VI*, 2010.

[Santana *et al.*, 2016] Pedro Santana, Sylvie Thiébaux, and Brian Williams. RAO*: An Algorithm for Chance-Constrained POMDPs. In *AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[Schrittwieser *et al.*, 2020] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839), 2020.

[Silver *et al.*, 2016] David Silver, Aja Huang, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 2016.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676), 2017.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), 2018.

[Sokota *et al.*, 2021] Samuel Sokota, Caleb Y. Ho, et al. Monte Carlo Tree Search with Iteratively Refining State Abstractions. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:18698–18709, 2021.

[Soualhia *et al.*, 2020] Mbarka Soualhia, Foutse Khomh, and Sofiène Tahar. A Dynamic and Failure-Aware Task Scheduling Framework for Hadoop. *IEEE Transactions on Cloud Computing*, 8(2):553–569, 2020.

[Sunberg and Kochenderfer, 2018] Zachary N. Sunberg and Mykel J. Kochenderfer. Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 28, 2018.

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

[Wan and Van Der Merwe, 2000] Eric A. Wan and Rudolph Van Der Merwe. The Unscented Kalman Filter for Nonlinear Estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*, pages 153–158. IEEE, 2000.