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

Robert J. Moss¹, Arec Jamgochian¹, Johannes Fischer¹,², Anthony Corso¹, and Mykel J. Kochenderfer¹
¹Stanford University, Stanford, CA
²Karlsruhe 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 CO₂ 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.

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 policy 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 $\Delta$-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]. $\Delta$-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 $\Delta$-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 $\Delta$-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 CO₂ 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 $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, \gamma)$ 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 $\pi$:

$$\pi(b_0) = \arg \max_{a \in \mathcal{A}} \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R_b(b_t, a_t) \middle| b_0 \right]$$

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

$$R_b(b, a) = \int_{s \in \mathcal{S}} b(s) R(s, a) \, ds.$$  

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 $b' \sim T_b(\cdot | b, a)$ 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 $s' \sim T(\cdot | s, a)$. Then an observation is sampled from the observation model $o \sim O(\cdot | a, s')$ and finally the belief is updated to get the posterior

$$b'(s') \propto O(o | a, s') \int_{s \in \mathcal{S}} T(s' | s, a) b(s) \, ds.$$  

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 $(\mathcal{B}, \mathcal{A}, T_b, R_b, \gamma)$ 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 $(\mathcal{S}, \mathcal{A}, O, G, \gamma)$ and the generative BMDP $(\mathcal{B}, \mathcal{A}, G_b, \gamma)$.

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 $\Delta$:

$$\text{maximize } \pi \quad V^\pi(b_0) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R_b(b_t, a_t) \middle| b_0 \right]$$

subject to $F^\pi(b_0) = \mathbb{P}_\pi \left[ \sum_{t=0}^{\infty} \mathbf{1}(s_t, a_t) \in \mathcal{F} \middle| b_0 \right] \leq \Delta$

The failure probability $F^\pi(b_0)$ is often called the execution risk of the policy $\pi$ computed from the belief $b_0$.

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

$$F_b(b, a) = \int_{s \in \mathcal{S}} b(s) \mathbf{1}\{ (s, a) \in \mathcal{F} \} \, ds$$

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

---

1https://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 $\tilde{b}$). 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. [2021] 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 uses UCT with a table-based value and risk predictor, denoted $\phi(b) = (\mu, \sigma)$ of summary statistics, and estimates the value $v = V_\phi(\tilde{b})$ and policy vector $p = P_\phi(\tilde{b})$ as two heads of the network $(p, v) = f_\phi(\tilde{b})$. 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:

$$\pi_{rew}(b, a) \propto \left( \sum_{a'} \exp Q(b, a') \right)^{1/\tau} \left( \sum_{a'} N(b, a') \right)$$

where $\tau$ controls the sampling temperature (used during policy iteration) and returns the argmax when $\tau = 0$ (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 $\Delta$-MCTS executions are run and a data set $D$ is collected. The data set $D = \{\{b_t, \pi_t, g_t, e_t\}_{t=1}^T\}_{j=1}^n$ is a tuple of the belief at episode time step $t$ denoted $b_t$, the tree policy $\pi_t$, the return $g_t = \sum_{t=1}^T \gamma^{(t-t)} r_i$ based on the observed reward $r_i$, and discount factor $\gamma$, and the failure event indicator $e_t$, where $g_t$ and $e_t$ are computed at the end of the trajectory for all time $t \leq T$. 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:

$$e_t = \mathbb{1}\{\bigwedge_{i=t}^T (s_i, a_i) \in \mathcal{F}\}$$

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: $P_{cc} \equiv (S, A, O, F, G, \tau, D_0)$ \(\triangleright\) Generative CC-POMDP

Require: $\psi$: Parameters

1. function PolicyIteration($P_{cc}, \psi$)
   2. \(f_0 \leftarrow \text{INITIALIZENETWORK}(\psi)\) \(\triangleright\) (\(P_0, V_0, F_0\) \(\leftarrow f_0\)
   3. for \(t \leftarrow 1\) to \(n_{\text{iterations}}\)
      4. \(D \leftarrow \text{COLLECTDATA}(P_{cc}, f_0)\) \(\triangleright\) policy evaluation
      5. \(f_0 \leftarrow \text{TRAIN}(f_0, D)\) \(\triangleright\) policy improvement
   6. return $\pi(b) \leftarrow \Delta\text{MCTS}(P_{cc}, f_0, b)$ \(\triangleright\) online policy (alg. 2)

1. function CollectData($P_{cc}, f_0$)
   2. $D = ()$
   3. parallel for \(j \leftarrow 1\) to \(n_{\text{data}}\)
      4. for \(t \leftarrow 1\) to \(T\)
         5. $a_t \leftarrow \Delta\text{MCTS}(P_{cc}, f_0, b_t)$ \(\triangleright\) sample a trajectory
         6. $s_{t+1}, r_t, a_t \sim G(s_t, a_t)$ \(\triangleright\) generative model
         7. $b_{t+1} \leftarrow P(s_{t+1} \mid b_t, a_t, a_t) \triangleright\) update belief
      8. for \(t \leftarrow 1\) to \(T\)
         9. $e_t \leftarrow \sum_{i \in T} g(i-t)_{t_i} \triangleright\) computed over trajectory
        10. $\mathcal{D}_j^{(t)} \leftarrow \mathcal{D}_j^{(t)} \cup \{(b_t, \pi_{\text{true}}(s_t, e_t))\}$ \(\triangleright\) failure event
   11. return $D$

During policy improvement, the neural network is trained to minimize the MSE or MAE loss $\mathcal{L}_{V_0}(g_t, v_t)$ to regress the value function $v_t = V_0(b_t)$, which mimics the cross-entropy loss $\mathcal{L}_{P_0}(\pi_t, p_t)$ to imitate the tree policy $p_t = F_0(b_t)$, and additionally minimizes the binary cross-entropy loss $\mathcal{L}_{F_0}(e_t, p_t)$ to regress the failure probability function $p_t = F_0(b_t)$, with added regularization using the $L_2$-norm of the weights $\theta$:

$$
\mathcal{L}_{V_0}(g_t, v_t) = (g_t - v_t)^2 \quad \text{or} \quad |g_t - v_t|
$$

$$
\mathcal{L}_{P_0}(\pi_t, p_t) = -\pi_t^T \log p_t
$$

$$
\mathcal{L}_{F_0}(e_t, p_t) = -e_t \log p_t - (1 - e_t) \log(1 - p_t)
$$

$$
\ell_{\text{CZ}} = \mathcal{L}_{V_0}(g_t, v_t) + \mathcal{L}_{P_0}(\pi_t, p_t) + \mathcal{L}_{F_0}(e_t, p_t) + \lambda ||\theta||^2
$$

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 $\Delta$-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 $p = F_0(b, a)$ 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 $p' = F_0(b')$. 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:

$$
P(E_{t:T}) = P(E) + P(E') - P(E \cap E') \quad (9)
$$

$$
P(E) = P(E) - P(E')P(E | E') \quad (10)
$$

$$
P(E) = P(E) + P(E') - P(E')P(E) \quad (11)
$$

$$
p = p + p' - pp' \quad (12)
$$

$$
p = p + (1 - p)p' \quad (13)
$$

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

$$
p = p + \delta(1 - p)p' \quad (14)
$$

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

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

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 $F_0(b, a)$, or the bootstrapped value by taking action $a$ to get a new belief $b'$ and computing eq. (14) based on the $p' = F_0(b')$ 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 $\Delta$ would be to use the PUCT algorithm [Silver et al., 2018]:

$$
\pi_{\text{exploration}}(b) = \arg \max_{a \in A(b)} Q(b, a) + c \left( \frac{P_0(b, a) \sqrt{\ln N(b, a)} + \Delta}{N(b, a)} \right) \quad (16)
$$

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 $P_0$ to explore promising actions.5

However, if the failure probability threshold $\Delta$ is too conservative, the action-selection process may fail to find any action that satisfies the constraint. Therefore, $\Delta$-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.

5The 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 $c$:

$$
\bar{Q}(b, a) = \frac{Q(b, a) - \min_{(b', a') \in T} Q(b', a')} {\max_{(b', a') \in T} Q(b', a') - \min_{(b', a') \in T} Q(b', a')} \quad (17)
$$
Figure 2: ConstrainedZero online Monte Carlo tree search with failure threshold adaptation (Δ-MCTS).

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

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

(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 $F(b, a) \leq \Delta'(b)$ where the selection threshold $\Delta'(b) = \max(\Delta_0, \Delta(b))$ upper bounds the failure probability. Together, the $\Delta$-MCTS exploration policy becomes:

$$\pi_{\text{explore}}(b) = \arg \max_{a \in A(b)} \bar{Q}(b, a) + c\left(\tilde{P}_b(b, a) \sqrt{\frac{N(b,a)}{1 + N(b,a)}}\right)$$

(21)

subject to $F(b, a) \leq \Delta'(b)$

(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, $\pi_{\text{explore}}$ is computed using the indicator $I$, returning the action $a \in A(b)$ that maximizes:

$$I \{ F(b, a) \leq \Delta'(b) \} \left( \bar{Q}(b, a) + c\left(\tilde{P}_b(b, a) \sqrt{\frac{N(b,a)}{1 + N(b,a)}}\right) \right)$$

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 $\Delta$-MCTS are shown in algorithms 2–3: selection, expansion, simulation, backpropagation, and adaptation, with extensions to BetaZero shown in red.

## 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 $R_b(b, a) = R_\text{rel}(b, a) - \lambda C(b, a)$ with a cost $C$ scaled by $\lambda$. 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 ownership 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 ownership is within 50 meters in relative altitude $h_{\text{rel}}$ of the intruder at the end of the encounter (i.e., when $|h_{\text{rel}}| \leq 50$ 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 $\lambda$. 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

---

**Figure 3:** BetaZero(\(\lambda\)) comparison.

**Figure 4:** Results for the collision avoidance CC-POMDP. Figure 4d matches the “notch” behavior from Kochenderfer et al. [2012].
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 ∆-MCTS without network approximators against ConstrainedZero, it is clear that offline policy iteration allows for better online planning. Using only the raw policy head $P_0$ 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 $\Delta_0$.

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 ∆-MCTS algorithm modifies MCTS for CC-POMDPs and addresses the issue of overfitting to failure predictions. Additionally, a constrained action-selection criterion (CC-UCT) was developed to enable planning under constraints. The full implementation and experiments are open sourced and are part of the Julia package BetaZero.jl.

#### 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 ∆-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 ∆-MCTS unexplored in this work is that it also applies directly to safety-critical MDPs, such as scheduling [Souláhia 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].

---

**Table 1: ConstrainedZero results.** Bold indicates the best results within the $\Delta_0$ threshold.

<table>
<thead>
<tr>
<th>LightDark</th>
<th>Collision Avoidance</th>
<th>Spillpoint CCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta_0 = 0.01$</td>
<td>$\Delta_0 = 0.01$</td>
<td>$\Delta_0 = 0.05$</td>
</tr>
<tr>
<td>$p(\text{fail})$</td>
<td>Returns ↑</td>
<td>$p(\text{fail})$</td>
</tr>
<tr>
<td>ConstrainedZero</td>
<td>0.01±0.01</td>
<td>13.07±0.42</td>
</tr>
<tr>
<td>No Adaptation</td>
<td>0.66±0.05</td>
<td>27.47±3.90</td>
</tr>
<tr>
<td>∆-MCTS (no $f_\beta$)</td>
<td>0.01±0.01</td>
<td>1.86±0.20</td>
</tr>
<tr>
<td>Raw Policy $P_0$</td>
<td>0.01±0.01</td>
<td>12.88±0.46</td>
</tr>
<tr>
<td>Raw Value $V_0$</td>
<td>0.72±0.05</td>
<td>28.00±4.51</td>
</tr>
<tr>
<td>Raw Failure $F_0$</td>
<td>0.80±0.04</td>
<td>0.05±0.04</td>
</tr>
</tbody>
</table>

All results report the mean ± standard error over 100 seeds, evaluated using the argmax of eq. (7), i.e., $\tau \rightarrow 0$.

† Trained with the same parameters as ConstrainedZero without adaptation, i.e., only a hard constraint on $\Delta_0$.

‡ ∆-MCTS without the neural network for the value or failure probability and a random policy for CC-UCT.

† One-step look-ahead over all actions using only the value or failure probability network head with 5 obs. per action.
Acknowledgments
This research is funded by OMV and J.F. thanks the Karlsruhe House of Young Scientists (KHYS) for travel grant funding.

References


