# Monte-Carlo Tree Search for Constrained POMDPs

**Jongmin Lee**[1], **Geon-Hyeong Kim**[1], **Pascal Poupart**[2], **Kee-Eung Kim**[1,3]
[1] School of Computing, KAIST, Republic of Korea
[2] University of Waterloo, Waterloo AI Institute and Vector Institute
[3] PROWLER.io
{jmlee,ghkim}@ai.kaist.ac.kr, ppoupart@uwaterloo.ca, kekim@cs.kaist.ac.kr

## Abstract

Monte-Carlo Tree Search (MCTS) has been successfully applied to very large POMDPs, a standard model for stochastic sequential decision-making problems. However, many real-world problems inherently have multiple goals, where multi-objective formulations are more natural. The constrained POMDP (CPOMDP) is such a model that maximizes the reward while constraining the cost, extending the standard POMDP model. To date, solution methods for CPOMDPs assume an explicit model of the environment, and thus are hardly applicable to large-scale real-world problems. In this paper, we present CC-POMCP (Cost-Constrained POMCP), an online MCTS algorithm for large CPOMDPs that leverages the optimization of LP-induced parameters and only requires a black-box simulator of the environment. In the experiments, we demonstrate that CC-POMCP converges to the optimal stochastic action selection in CPOMDP and pushes the state-of-the-art by being able to scale to very large problems.

## 1 Introduction

Monte-Carlo Tree Search (MCTS) [4, 5, 12] is a generic online planning algorithm that effectively combines random sampling and tree search, and has shown great promise in many areas such as online Bayesian reinforcement learning [8, 10] and computer Go [7, 20]. MCTS efficiently explores the search space by investing more search effort in promising states and actions while balancing exploration and exploitation in the direction of maximizing the cumulative (scalar) rewards. Due to its outstanding performance without relying on any prior domain knowledge or heuristic function, MCTS has become the de-facto standard method for solving very large sequential decision making problems, commonly formulated as Markov decision processes (MDPs) and partially observable MDPs (POMDPs).

However in many situations, it is not straightforward to formulate the objective with the reward maximization alone, as in the following examples. For spoken dialogue systems [24], it is common to optimize the dialogue strategy towards minimizing the number of turns while maintaining the success rate of dialogue tasks above a certain level. For UAVs under search and rescue mission, the main goal would be to find as many targets as possible, while avoiding threats that may endanger the mission itself. The constrained POMDP (CPOMDP) [9] is an appealing framework for dealing with this kind of multi-objective sequential decision making problems when the environment is partially observable. The model assumes that the action incurs not only rewards, but also $K$ different types of costs, and the goal is to find an optimal policy that maximizes the expected cumulative rewards while bounding each of $K$ expected cumulative costs below certain levels.

Although the CPOMDP is a very expressive model, it is known to be very difficult to solve due to the PSPACE-complete nature of solving POMDP [16] originating from the two main challenges: the *curse of dimensionality* and the *curse of history*. Partially observable Monte-Carlo Planner (POMCP) [19] tames these two curses of POMDP by using Monte-Carlo sampling both in the root belief-state

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

and in the black-box simulation of the history. In contrast, solution methods for CPOMDPs, e.g. dynamic programming [11], linear programming [18], and online uniform-cost search [23], are not yet advanced up to this level, partly due to requiring an explicit model of the environment. This prevents CPOMDP from being a practical approach for modeling real-world applications.

In this paper, we present an MCTS algorithm for CPOMDPs, which precisely addresses the scalability. To the best of our knowledge, extending MCTS to CPOMDPs (or even CMDPs) has remained unexplored since it is not straightforward to handle the constrained optimization in tree search. This challenge is compounded by the fact that optimal policies can be stochastic.[1] In order to develop MCTS for CPOMDPs, we first show that solving CPOMDPs is essentially equivalent to jointly solving an unconstrained POMDP while optimizing its LP-induced parameters that control the trade-off between the reward and the costs. From this result, we present our algorithm, Cost-Constrained POMCP (CC-POMCP), for solving large CPOMDPs that combine traditional MCTS with LP-induced parameter optimization. In the experiments section, we demonstrate that CC-POMCP converges to the optimal stochastic action selection using a synthetic domain and that it is able to handle very large problems including constrained variants of Rocksample(15,15) and Atari 2600 arcade game, pushing the state-of-the-art scalability in CPOMDP solvers.

## 2 Background

Partially observable Markov decision processes (POMDPs) [22] provide a principled framework for modeling sequential decision making problems under stochastic transitions and noisy observations. It is formally defined by tuple $\langle S_p, A, O_p, T_p, Z_p, R_p, \gamma, b_0 \rangle$, where $S_p$ is the set of environment states $s$, $A$ is the set of actions $a$, $O_p$ is the set of observations $o$, $T_p(s'|s,a) = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$ is the transition probability, $Z_p(o|s',a) = \Pr(o_{t+1} = o|s_{t+1} = s', a_t = a)$ is the observation probability, $R_p(s,a) \in \mathbb{R}$ is the immediate reward for taking action $a$ in state $s$, $\gamma \in [0,1)$ is the discount factor, and $b_0(s) = \Pr(s_0 = s)$ is the starting state distribution at time step 0. The history $h_t = [a_0, o_0, \ldots, a_t, o_t]$ and $h_t a_{t+1} = [a_0, o_0, \ldots, a_t, o_t, a_{t+1}]$ denote the sequence of actions and observations. The agent takes an action via policy $\pi(a|h) = \Pr(a_t = a|h_t = h)$ that maps from history to probability distribution over actions. In POMDPs, the environment state is not directly observable, thus the agent maintains belief $b_t(s) = \Pr(s_t = s|h_t)$ that can be recursively updated using the Bayes rule: when taking action $a$ in $b$ and observing $o$, the updated belief is $b^{ao}(s') \propto Z_p(o|s',a) \sum_s T_p(s'|s,a) b(s)$. Since belief $b_t$ is a sufficient statistic of history $h_t$, the POMDP can be understood as the belief-state MDP $\langle B, A, T, R, \gamma, b_0 \rangle$, where $b_0$ is the initial state, $B$ is the set of reachable beliefs starting from $b_0$, $T(b'|b,a) = \sum_{o,s,s'} Z_p(o|s',a) T_p(s'|s,a) b(s) \delta(b', b^{ao})$ is the transition probability, $R(b,a) = \sum_s b(s) R_p(s,a)$ is the immediate reward function. We shall use $h$ and $b = \Pr(s|h)$ interchangeably as long as there is no confusion (e.g. $Q_R(h,a) = Q_R(b,a)$, $\pi(a|h) = \pi(a|b)$). The goal is to find an optimal policy $\pi^*$ that maximizes the expected discounted return (i.e. cumulative discounted rewards):

$$\max_\pi V_R^\pi(b_0) = E_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(b_t, a_t) \;\middle|\; b_0 \right].$$

Constrained POMDPs (CPOMDPs) [9, 11, 18] is a generalization of POMDPs for multi-objective problems. It is formally defined by tuple $\langle S_p, A, O_p, T_p, Z_p, R_p, \mathbf{C}_p, \hat{\mathbf{c}}, \gamma, b_0 \rangle$, where $\mathbf{C}_p = \{C_{p,k}\}_{1..K}$ is the set of $K$ non-negative cost functions with individual thresholds $\hat{\mathbf{c}} = \{\hat{c}_k\}_{1..K}$. Similarly, a CPOMDP can be cast into an equivalent belief-state CMDP $\langle B, A, T, R, \mathbf{C}, \hat{\mathbf{c}}, \gamma \rangle$, where $C_k(b,a) = \sum_s b(s) C_{p,k}(s,a)$. The goal is to compute an optimal policy that maximizes the expected cumulative reward while bounding the expected cumulative costs:

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

$$s.t. \; V_{C_k}^\pi(b_0) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t C_k(b_t, a_t) \;\middle|\; b_0 \right] \le \hat{c}_k \;\; \forall k$$

[1] Stochastic nature of the optimal policy in CPOMDPs results from the stochasticity of optimal policies in CMDPs [1]. A more formal treatment on this matter, pertinent to CPOMDPs, can be found in [6].

An optimal policy of the CPOMDP (or the equivalent belief-state CMDP) is generally stochastic and can be obtained by solving the following linear program (LP) [1]:

$$\begin{aligned}
&\max_{\{y(b,a)\geq 0\}\forall b,a} \sum_{b,a} R(b,a)y(b,a) \qquad (1)\\
&s.t. \ \sum_{a'} y(b',a') = \delta(b_0,b') + \gamma \sum_{b,a} T(b'|b,a)y(b,a) \ \ \forall b'\\
&\qquad \sum_{b,a} C_k(b,a)y(b,a) \leq \hat{c}_k \ \ \ \forall k
\end{aligned}$$

where $y(b,a)$ can be interpreted as a discounted occupancy measure of $(b,a)$, and $\delta(x,y)$ is a Dirac delta function that has the value of 1 if $x = y$ and 0 otherwise. Once the optimal solution $y^*(b,a)$ is obtained, an optimal stochastic policy and the corresponding optimal value are computed by $\pi^*(a|b) = \Pr(a|b) = y^*(b,a)/\sum_{a'} y^*(b,a')$ and $V_R^*(b_0;\hat{\mathbf{c}}) = \sum_{b,a} R(b,a)y^*(b,a)$ respectively. It is usually intractable to solve LP (1) exactly since the cardinality of $B$ can be infinite.

POMCP [19] is a highly scalable Monte-Carlo tree search (MCTS) algorithm for (unconstrained) POMDPs. The algorithm uses Monte-Carlo simulation for both tree search and belief update to simultaneously tackle the *curse of history* and the *curse of dimensionality*. In each simulation, a state particle is sampled from the root node's belief-state $s \sim \mathcal{B}(h)$ (called root sampling) and is used to sample a trajectory using a black-box simulator $(s', o, r) \sim \mathcal{G}(s,a)$. It adopts UCB1 [2] as the tree policy, i.e. the action selection rule in the internal nodes of the search tree:

$$\arg\max_a \left[ Q_R(h,a) + \kappa \sqrt{\frac{\log N(h)}{N(h,a)}} \right]$$

where $Q_R(h,a)$ is the average of the sampled returns, $N(h)$ is the number of simulations performed through $h$, $N(h,a)$ is the number of times action $a$ is selected in $h$, and $\kappa$ is the exploration constant to adjust the exploration-exploitation trade-off. POMCP expands the search tree non-uniformly, focusing more search efforts in promising nodes. It can be formally shown that $Q_R(h,a)$ asymptotically converges to the optimal value $Q_R^*(h,a)$ in POMDPs.

Unfortunately, it is not straightforward to use POMCP for CPOMDPs since the original UCB1 action selection rule does not have any notion of cost constraints. If we naively adopt the vanilla, reward-maximizing POMCP, we may obtain cost-violating action sequences. We could obtain the average of sampled cumulative costs $Q_C$ during search, but it is not straightforward how to leverage them in the tree policy: if we naively prevent action branches that violate the cost constraint $Q_C(h,a) \leq \hat{c}$, we may end up with policies that are too conservative and thus sub-optimal, i.e. a feasible policy may be rejected during search if the Monte-Carlo estimate violates the cost constraint.

## 3 Solving CPOMDP via a POMDP Solver

The derivation of our algorithm starts from the dual of (1):

$$\begin{aligned}
&\min_{\substack{\{\mathcal{V}(b)\}\forall b\\ \{\lambda_k\geq 0\}\forall k}} \sum_b \delta(b_0,b)\mathcal{V}(b) + \sum_k \hat{c}_k\lambda_k \qquad (2)\\
&s.t. \quad \mathcal{V}(b) \geq R(b,a) - \sum_k C_k(b,a)\lambda_k + \gamma \sum_{b'} T(b'|b,a)\mathcal{V}(b') \ \ \ \forall b,a
\end{aligned}$$

Observe that if we treat $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_K]^\top$ as a constant, the problem becomes an *unconstrained* belief-state MDP with the scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. Let $\mathcal{V}_{\boldsymbol{\lambda}}^*$ be the optimal value function of this unconstrained POMDP. Then, for any $\boldsymbol{\lambda}$, there exists a corresponding unique $\mathcal{V}_{\boldsymbol{\lambda}}^*$, and we can compute $\mathcal{V}_{\boldsymbol{\lambda}}^*$ with a POMDP solver. Thus, solving the dual LP (2) reduces to:

$$\min_{\boldsymbol{\lambda}\geq\mathbf{0}} \left[ \mathcal{V}_{\boldsymbol{\lambda}}^*(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}} \right] \qquad (3)$$

Moreover, if there is an optimal solution $y^*$ to the primal LP in (1), then there exists a corresponding dual optimal solution $\mathcal{V}^*$ and $\boldsymbol{\lambda}^*$, and the duality gap is zero, i.e.

$$V_R^*(b_0;\hat{\mathbf{c}}) = \sum_{b,a} R(b,a)y^*(b,a) = \mathcal{V}_{\boldsymbol{\lambda}^*}^*(b_0) + \boldsymbol{\lambda}^{*\top}\hat{\mathbf{c}}$$

by the strong duality theorem.

To compute optimal $\boldsymbol{\lambda}$ in Eq. (3), we have to consider the trade-off between the first term and the second term according to the cost constraint $\hat{\mathbf{c}}$. For example, if the cost constraint $\hat{\mathbf{c}}$ is very large, the optimal solution $\boldsymbol{\lambda}^*$ tends to be close to zero since the objective function would be mostly affected by the second term $\boldsymbol{\lambda}^\top \hat{\mathbf{c}}$. On the other hand, if $\hat{\mathbf{c}}$ is sufficiently small, the first term will be dominant and the optimal solution $\boldsymbol{\lambda}^*$ tends to get larger in order to have a negative impact on the reward $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. Thus, it may seem that Eq. (3) is a complex optimization problem. However, as we will see in the following proposition, the objective function in Eq. (3) is actually piecewise-linear and convex over $\boldsymbol{\lambda}$, as depicted in Figure 3 in Appendix A.

**Proposition 1.** *Let $\mathcal{V}^*_{\boldsymbol{\lambda}}$ be the optimal value function of the POMDP with scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. Then, $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}}$ is a piecewise-linear and convex (PWLC) function of $\boldsymbol{\lambda}$. (The proof is provided in Appendix A.)*

In addition, we can show that the optimal solution $\boldsymbol{\lambda}^*$ is bounded:

**Proposition 2** (Lemma 4 in [14])**.** *Suppose that the reward function is bounded in $[R_{\min}, R_{\max}]$ and there exists $\tau > 0$ and a (feasible) policy $\pi$ such that $V^\pi_{\mathbf{C}}(b_0) + \tau \mathbf{1} \leq \hat{\mathbf{c}}$. Then, $\|\boldsymbol{\lambda}^*\|_1 \leq \frac{R_{\max} - R_{\min}}{\tau(1-\gamma)}$.*

Thus, from Propositions 1 and 2, we can obtain optimal $\boldsymbol{\lambda}^*$ by greedily optimizing (3) with $\lambda_k$ in the range $[0, \frac{R_{\max} - R_{\min}}{\tau(1-\gamma)}]$. The remaining question is how to compute that direction for updating $\boldsymbol{\lambda}$. We start with the following lemma to answer this question:

**Lemma 1.** *Let $\mathcal{M}_1 = \langle B, A, T, R_1, \gamma \rangle$ and $\mathcal{M}_2 = \langle B, A, T, R_2, \gamma \rangle$ be two (belief-state) MDPs differing only in the reward function, and $V^\pi_1$ and $V^\pi_2$ be the value functions of $\mathcal{M}_1$ and $\mathcal{M}_2$ with a fixed policy $\pi$. Then, the value function of the new MDP $\mathcal{M} = \langle B, A, T, pR_1 + qR_2, \gamma \rangle$ with the policy $\pi$ is $V^\pi(b) = pV^\pi_1(b) + qV^\pi_2(b)$ for all $b \in B$. (The proof is provided in Appendix B.)*

Lemma 1 implies that $\mathcal{V}^*_{\boldsymbol{\lambda}}$ can be decomposed into $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) = V^{\pi^*_{\boldsymbol{\lambda}}}_R(b_0) - \boldsymbol{\lambda}^\top V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0)$ where $\pi^*_{\boldsymbol{\lambda}}$ is the optimal policy with respect to the scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$, and thus (3) becomes:

$$\min_{\boldsymbol{\lambda} \geq \mathbf{0}} \left[ V^{\pi^*_{\boldsymbol{\lambda}}}_R(b_0) - \boldsymbol{\lambda}^\top V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}} \right] \tag{4}$$

One way to compute the descent direction for $\boldsymbol{\lambda}$ would be by taking the derivative of Eq. (4) with respect to $\boldsymbol{\lambda}$ while holding $\pi^*_{\boldsymbol{\lambda}}$ constant so that we use the direction $V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) - \hat{\mathbf{c}}$. The following theorem shows that this is indeed a valid direction:

**Theorem 1.** *For any $\boldsymbol{\lambda}$, $V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) - \hat{\mathbf{c}}$ is a negative subgradient that decreases the objective in Eq. (3), where $\pi^*_{\boldsymbol{\lambda}}$ is the optimal policy with respect to the scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. Also, if $V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) - \hat{\mathbf{c}} = \mathbf{0}$ then $\boldsymbol{\lambda}$ is the optimal solution of Eq. (3). (Proof provided in Appendix C.)*

The direction $V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) - \hat{\mathbf{c}}$ has a natural interpretation: if the current policy violates the $k$-th cost constraint (i.e. $V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k} > \hat{c}_k$), $\lambda_k$ increases so that the cost is penalized more in the scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. On the other hand, if the current policy is too conservative for the $k$-th cost constraint (i.e. $V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k} < \hat{c}_k$), $\lambda_k$ decreases so that the cost is penalized less.

In summary, we can solve the dual of LP of the belief-state CMDP by iterating through the following steps, starting from any $\boldsymbol{\lambda}$:

1. $\pi^*_{\boldsymbol{\lambda}} \leftarrow$ SolveBeliefMDP($\langle B, A, T, R - \boldsymbol{\lambda}^\top \mathbf{C}, \gamma \rangle$)
2. $V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}} \leftarrow$ PolicyEvaluation($\langle B, A, T, \mathbf{C}, \gamma \rangle, \pi^*_{\boldsymbol{\lambda}}$)
3. $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha_n (V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) - \hat{\mathbf{c}})$ and clip $\lambda_k$ to range $[0, \frac{R_{\max} - R_{\min}}{\tau(1-\gamma)}] \quad \forall k \in \{1, 2, ...K\}$

By Theorem 1, this procedure is a subgradient method, guaranteed to converge to the optimal solution by using a step-size sequence $\alpha_n$ such that $\sum_n \alpha_n = \infty$ and $\sum_n \alpha_n^2 < \infty$.

**Algorithm 1** Cost-Constrained POMCP (CC-POMCP)

```
function SEARCH(h_0)
  λ is randomly initialized.
  repeat
    if h = ∅ then
      s ~ b_0
    else
      s ~ B(h_0)
    end if
    SIMULATE(s, h_0, 0)
    a ~ GREEDYPOLICY(h_0, 0, 0)
    λ ← λ + α_n [Q_C(h_0, a) − ĉ]
    Clip λ_k to range [0, (R_max − R_min)/(τ(1−γ))]   ∀k = {1, 2, ...K}
  until TIMEOUT()
  return GREEDYPOLICY(h_0, 0, ν)
end function

function ROLLOUT(s, h, d)
  if d = (maximum-depth) then
    return [0, 0]
  end if
  a ~ π_rollout(·|h)  and  (s', o, r, c) ~ G(s, a)
  return [r, c] + γ· ROLLOUT(s', hao, d + 1)
end function

function GREEDYPOLICY(h, κ, ν)
  Q^⊕_λ(h, a) := Q_R(h, a) − λ^⊤ Q_C(h, a) + κ sqrt(log N(h) / N(h, a))
  a* ← argmax_a Q^⊕_λ(h, a)
  A* ← { a*_i | |Q_λ(h, a*_i) − Q_λ(h, a*)|
                 ≤ ν ( sqrt(log N(h, a*_i) / N(h, a*_i)) + sqrt(log N(h, a*) / N(h, a*)) ) }
  Solve LP (10) with A* to compute a policy π(a*_i|h) = w_i.

  return π(·|h)
end function

function SIMULATE(s, h, d)
  if d = (maximum-depth) then
    return [0, 0]
  end if
  if h ∉ T then
    T(ha) ← (N_init, Q_{R,init}, Q_{C,init}, ∅) ∀a
    return ROLLOUT(s, h, d)
  end if
  a ~ GREEDYPOLICY(h, κ, ν)
  (s', o, r, c) ~ G(s, a)
  [R, C] ← [r, c] + γ· SIMULATE(s', hao, d + 1)
  B(h) ← B(h) ∪ {s}
  N(h) ← N(h) + 1
  N(h, a) ← N(h, a) + 1
  Q_R(h, a) ← Q_R(h, a) + (R − Q_R(h, a)) / N(h, a)
  Q_C(h, a) ← Q_C(h, a) + (C − Q_C(h, a)) / N(h, a)
  c̄(h, a) ← c̄(h, a) + (c − c̄(h, a)) / N(h, a)
  return [R, C]
end function

function MAINLOOP()
  ĉ ← (cost constraint)
  s ← (initial state)
  h ← ∅
  while s is not terminal do
    π ← SEARCH(h)
    a ~ π(·|h)
    (s', o, r, c) ~ G(s, a)
    ĉ ← (ĉ − π(a|h) c̄(h, a) − Σ_{a'≠a} π(a'|h) Q_C(h, a')) / (γ π(a|h))
    s ← s'
    h ← hao
  end while
end function
```

## 4 Cost-Constrained POMCP (CC-POMCP)

Although we have eliminated the cost-constraints by introducing simultaneous update of $\boldsymbol{\lambda}$, it still relies on exactly solving POMDPs via *SolveBeliefMDP* in each iteration, which is impractical for large CPOMDPs. Fortunately, all we need in step 3 is the cost value at the *initial* belief state $V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}}}(b_0)$ with respect to the optimal policy when the reward function is given by $R - \boldsymbol{\lambda}^\top \mathbf{C}$. This is exactly the situation where MCTS can be effectively applied: MCTS focuses on finding the optimal action selection at the *root node* using the Monte-Carlo estimate of long-term rewards (or costs). We are now ready to present our online algorithm for large CPOMDPs, which we refer to as Cost-Constrained POMCP (CC-POMCP), shown in Algorithm 1. The changes from the standard POMCP are highlighted in blue. CC-POMCP is an extension of POMCP with cost constraints and can be seen as an anytime approximation of policy iteration with the simultaneous optimization of $\boldsymbol{\lambda}$: the policy is sequentially evaluated via Monte-Carlo return

$$Q_R(h,a) \leftarrow Q_R(h,a) + \frac{R - Q_R(h,a)}{N(h,a)} \quad \text{and} \quad Q_{\mathbf{C}}(h,a) \leftarrow Q_{\mathbf{C}}(h,a) + \frac{\mathbf{C} - Q_{\mathbf{C}}(h,a)}{N(h,a)} \tag{5}$$

and the policy is implicitly improved by the UCB1 action selection rule based on the scalarized value $\mathcal{Q}_{\boldsymbol{\lambda}}(h,a) = Q_R(h,a) - \boldsymbol{\lambda}^\top Q_{\mathbf{C}}(h,a)$:

$$\arg\max_a \mathcal{Q}^{\oplus}_{\boldsymbol{\lambda}}(h,a) = \left[ Q_R(h,a) - \boldsymbol{\lambda}^\top Q_{\mathbf{C}}(h,a) + \kappa \sqrt{\frac{\log N(h)}{N(h,a)}} \right] \tag{6}$$

Finally, $\boldsymbol{\lambda}$ is updated simultaneously using the current estimate of $V_{\mathbf{C}}(s_0) - \hat{\mathbf{c}}$, which is the descent direction of the convex objective function:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha_n (Q_{\mathbf{C}}(h_0, a) - \hat{\mathbf{c}}) \text{ where } a \sim \pi(\cdot|h_0) \tag{7}$$

The following theorem states that CC-POMCP asymptotically converges to optimal $\boldsymbol{\lambda}^*$ under mild assumption:

**Theorem 2.** *Suppose that $\boldsymbol{\lambda}$ is updated with increasing simulation step $t$, and the search tree is reset at the end of $\boldsymbol{\lambda}$'s update as detailed in Appendix F. If the asymptotic bias of UCT holds for all types of cost values (i.e. $\exists M > 0$, $\forall k$, $|V_{C_k}^{\pi_{\boldsymbol{\lambda}}^*}(h_0) - V_{C_k}(h_0)| \leq M(\frac{\log t}{t})$), then either $sign(V_{C_k}^{\pi_{\boldsymbol{\lambda}}^*}(h_0) - \hat{c}_k) = sign(V_{C_k}(h_0) - \hat{c}_k)$ or $|V_{C_k}^{\pi_{\boldsymbol{\lambda}}^*}(h_0) - \hat{c}_k| \leq M \frac{\log t}{t}$ holds with probability 1 as $t \to \infty$.*

The above states that either $\boldsymbol{\lambda}$ is close to optimal or it is improved by the update towards the direction of negative subgradient. Note that CC-POMCP inherits the scalability of POMCP and thus does not require an explicit model of the environment: all we need is a black-box simulator $\mathcal{G}$ of the CPOMDP, which generates sample $(s', o, r, c) \sim \mathcal{G}(s, a)$ of the next state $s'$, observation $o$, reward $r$, and cost vector $c$, given the current state $s$ and action $a$.

### 4.1 Admissible Costs

After the agent executes an action, the cost constraint threshold $\hat{c}$ must be updated at the next time step. For this, we reformulate the notion of *admissible cost* [17], originally formulated for dynamic programming. The admissible cost $\hat{\mathbf{c}}_{t+1}$ at time step $t+1$ denotes the expected total cost allowed to be incurred in future time steps $\{t+1, t+2, ...\}$ without violating the cost constraints. Under dynamic programming, the update is given by [17]: $\hat{\mathbf{c}}_{t+1} = \frac{\hat{\mathbf{c}}_t - \mathbb{E}[\mathbf{C}(b_t, a_t)|b_0, \pi]}{\gamma}$ where evaluating $\mathbb{E}[\mathbf{C}(b_t, a_t)|b_0, \pi]$ requires the probability of reaching $(b_t, a_t)$ at time step $t$, which in turn requires marginalizing out the history in the past $[a_0, b_1, a_2, ..., b_{t-1}]$. This is intractable for large state spaces.

On the other hand, under forward search, the admissible cost at the next time step $t+1$ is simply $\hat{\mathbf{c}}_{t+1} = V_{\mathbf{C}}^{\pi^*}(b_{t+1})$. We can access $V_{\mathbf{C}}^{\pi^*}(b_{t+1})$ by starting from the root node of the search tree $h_t$, and sequentially following the action branch $a_t$ and the next observation branch $o_{t+1}$. Here we are assuming that the exact optimal $V_{\mathbf{C}}^{\pi^*}$ is obtained, which is certainly achievable after infinitely many simulations of CC-POMCP. Note also that even though $\hat{\mathbf{c}}_t > V_{\mathbf{C}}^{\pi^*}(b_t)$ is possible in general, assuming $\hat{\mathbf{c}}_t = V_{\mathbf{C}}^{\pi^*}(b_t)$ does not change the solution. If $\hat{\mathbf{c}}_t < V_{\mathbf{C}}^{\pi^*}(b_t)$, this means that no feasible policy exists.

### 4.2 Filling the Gap: Stochastic vs Deterministic Policies

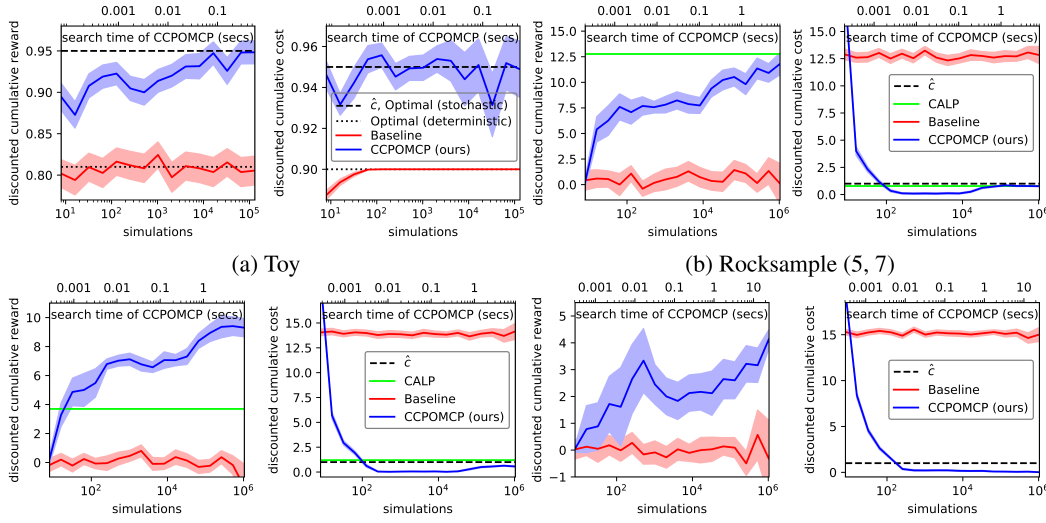
Our approach relies on the POMDP with scalarized rewards, but care must be taken as the optimal policy of the CPOMDP is generally stochastic: given optimal $\boldsymbol{\lambda}^*$, let $\pi_{\boldsymbol{\lambda}}^*$ be the *deterministic* optimal policy for the POMDP with the scalarized reward function $R - \boldsymbol{\lambda}^{*\top}\mathbf{C}$. Then, by the duality between the primal (1) and the dual (2),

$$V_R^*(b_0; \hat{\mathbf{c}}) = \mathcal{V}_{\lambda}^*(b_0) + \boldsymbol{\lambda}^{*\top}\hat{\mathbf{c}} = V_R^{\pi_{\boldsymbol{\lambda}}^*}(b_0) - \boldsymbol{\lambda}^{*\top}(V_{\mathbf{C}}^{\pi_{\boldsymbol{\lambda}}^*}(b_0) - \hat{\mathbf{c}}) \tag{8}$$

This implies that if $\lambda_k^* > 0$ and $V_{C_k}^{\pi_{\boldsymbol{\lambda}}^*}(b_0) \neq c_k$ for some $k$ then $\pi_{\boldsymbol{\lambda}}^*$ is not optimal for the original CPOMDP. This is exactly the situation where the optimal policy is stochastic. In order to make the policy computed by our algorithm stochastic, we make sure that the following *optimality condition* is satisfied, derived from $V_R^*(b_0; \hat{\mathbf{c}}) = V_R^{\pi_{\boldsymbol{\lambda}}^*}(b_0)$:

$$\sum_{k=1}^{K} \lambda_k^* (V_{C_k}^{\pi_{\boldsymbol{\lambda}}^*}(b_0) - \hat{c}_k) = \sum_{k=1}^{K} \lambda_k^* \left( \sum_a \pi_{\boldsymbol{\lambda}}^*(a|b_0) Q_{C_k}^{\pi_{\boldsymbol{\lambda}}^*}(b_0, a) - \hat{c}_k \right) = 0 \tag{9}$$

That is, actions $a_i^*$ with equally maximal scalarized action values $\mathcal{Q}_{\boldsymbol{\lambda}}(b, a_i^*) = Q_R(b, a_i^*) - \boldsymbol{\lambda}^\top Q_{\mathbf{C}}(b, a_i^*)$ participate as the support of the stochastic policy, and are selected with probability $\pi(a_i^*|b)$ that satisfies $\forall k : \lambda_k^* > 0,\ \sum_{a_i^*} \pi(a_i^*|b) Q_{C_k}(b, a_i^*) = \hat{c}_k$.




(a) Toy (b) Rocksample (5, 7)

(c) Rocksample (7, 8) (d) Rocksample (11, 11)

Figure 1: The result of CPOMDP Toy domain [11] and the constrained variants of *Rocksample* [21]. The result of *Rocksample* (15, 15) is presented in Appendix I. For each domain, the left figure shows the average discounted cumulative reward, and the right figure shows the average discounted cumulative cost. The wall-clock search time for CC-POMCP is presented on the top of $x$-axis.

GreedyPolicy in Algorithm 1 computes the stochastic policy according to the above principle. In practice, due to the randomness in Monte-Carlo sampling, action values in (9) are always subject to estimation error, so it is reformulated as a linear programming with up to $|A| + 2K$ variables:

$$\begin{aligned} &\min_{\{w_i, \xi_k^+, \xi_k^-\}} \sum_{k=1}^{K} \lambda_k (\xi_k^+ + \xi_k^-) \qquad (10) \\ &s.t. \sum_{i: a_i^* \in A^*} w_i Q_{C_k}(h, a_i^*) = \hat{c}_k + (\xi_k^+ - \xi_k^-) \quad \forall k \\ &\sum_{i: a_i^* \in A^*} w_i = 1 \text{ and } w_i, \xi_k^+, \xi_k^- \geq 0 \end{aligned}$$

where $A^* = \{a_i^* \mid \mathcal{Q}_{\boldsymbol{\lambda}}(h, a_i^*) \simeq \mathcal{Q}_{\boldsymbol{\lambda}}(h, a^*) \text{ s.t. } a^* = \arg\max_a \mathcal{Q}_{\boldsymbol{\lambda}}^{\oplus}(h, a)\}$ [2], and the solutions are $w_i = \pi(a_i^*|h)$. Here, when $K = 1$, there is a simple analytic solution to LP (10), which is described in Appendix G. Even when $K > 1$, note that the optimization problem occurs only when the number of equally maximal scalarized action values is more than 1 thus randomization of actions is required. It is well known in CMDPs that an optimal policy requires at most $K$ randomizations [1], which means that we expect to invoke optimization on extremely small part of the state space when the problem is very large.

## 5 Experiments

All the parameters for running CC-POMCP are provided in Appendix H.

**Baseline agent** To the best of our knowledge, this work is the first attempt to solve *constrained* (PO)MDP using Monte-Carlo Tree Search. Since there is no algorithm for direct performance comparison for large problems, we implemented a simple baseline agent using MCTS. This agent works as outlined in section 2: it chooses an action via reward-maximizing POMCP while preventing action branches that violate cost constraint $Q_C(s, a) \leq \hat{c}$. If all action branches violate the cost constraints, the agent chooses action uniformly at random.

[2]Exact condition for $\mathcal{Q}_{\boldsymbol{\lambda}}(h, a_i^*) \simeq \mathcal{Q}_{\boldsymbol{\lambda}}(h, a^*)$ and its theoretical guarantee are provided in Appendix E.

| Domain | $\lvert S\rvert$ | $\hat{c}$ | Algorithm | Cumulative reward | Cumulative cost |
|---|---|---|---|---|---|
| Rocksample (5,7) | 3,201 | 1 | CALP | **12.77±0** | **0.78±0** |
| | | | Baseline | 1.09±0.88 | 12.74±0.50 |
| | | | CC-POMCP | 11.36±1.02 | 0.79±0.06 |
| Rocksample (7,8) | 12,544 | 1 | CALP | 3.67±0 | 1.20±0 |
| | | | Baseline | −0.23±0.44 | 13.92±0.33 |
| | | | CC-POMCP | **9.36±0.76** | **0.56±0.06** |
| Rocksample (11,11) | 247,808 | 1 | CALP | N/A | N/A |
| | | | Baseline | 0.14±0.33 | 15.29±0.25 |
| | | | CC-POMCP | **2.65±0.73** | **0.09±0.04** |
| Rocksample (15,15) | 7,372,800 | 1 | CALP | N/A | N/A |
| | | | Baseline | 0.39±0.58 | 16.27±0.27 |
| | | | CC-POMCP | **0.74±0.33** | **0.69±0.08** |

Table 1: Comparison of CC-POMDP with the state-of-the-art offline solver, CALP [18].

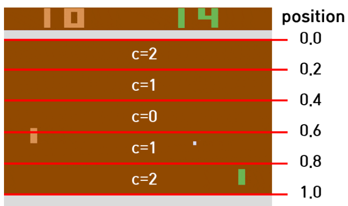
**CPOMDP: Toy and Rocksample** We first tested CC-POMCP on the synthetic toy domain introduced in [11] to demonstrate convergence to stochastic optimal actions, where the cost constraint $\hat{c}$ is 0.95. Any deterministic policy is suboptimal or violates the cost constraint. As can be seen in Figure 1a, CC-POMCP converges to optimal stochastic action selection (thus experimentally confirms the soundness of algorithm), while the baseline agent converges to the suboptimal policy (optimal policy among deterministic ones).

We also conducted experiments on cost-constrained variants of *Rocksample* [21]. *Rocksample*$(n, k)$ simulates a Mars rover in $n \times n$ grid containing $k$ rocks. The goal is to sort out good rocks, collect them, and escape the map by moving to the rightmost part of the map. We augmented the single-objective *Rocksample* with the cost function that assigns 1 to low reward state-action pairs (i.e. $C_p(s, a) = 1$ if $R_p(s, a) < 0$), similarly to [18]. We also assigned the cost of 1 to actions detecting whether a rock is good or bad. The cost constraint $\hat{c}$ is set to 1. We compared CC-POMCP with the state-of-the-art offline CPOMDP solver, CALP [18]. CALP was allowed 10 minutes for the offline computation, and we performed exact policy evaluation with respect to the resulting finite state controller without simulation in the real environment. The results on *Rocksample* are summarized in Table 1 and Figure 1. In *Rocksample* (5, 7), the reward performance of CC-POMCP is comparable to CALP when more than 2 seconds of search time is allowed while at the same time satisfying the cost constraint. In contrast, baseline agent basically exhibited random behavior since the Monte-Carlo return at early stage mostly violates cost constraints for all actions. On *Rocksample (7, 8)*, CALP failed to compute a feasible policy, and CC-POMCP outperformed CALP in terms of reward while satisfying the cost constraint. Finally, CC-POMCP was able to scale to *Rocksample (11, 11)* and *(15, 15)*: given a few seconds of search time, CC-POMCP was able to find actions satisfying the cost constraints, and tended to yield higher returns as we increased the number of simulations.

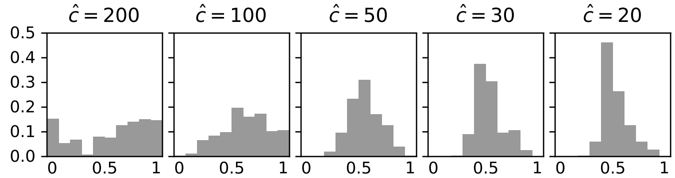
**CMDP: Pong** We also conducted experiments on a multi-objective version of PONG, an arcade game running on the Arcade Learning Environment (ALE) [3], depicted in Figure 2a. In this domain, the left paddle is handled by the default computer opponent and the right paddle is controlled by the agent. We use the RAM state feature, i.e. the states are binary strings of length 1024 which results in $|S| = 2^{1024}$. The action space is {up, down, stay}. The agent receives a reward of $\{1, -1\}$ for each round depending on win/lose. The episode terminates if the accumulated reward is $\{21, -21\}$. We assigned cost 0 to the center area ($position \in [0.4, 0.6]$), 1 to the neighboring area ($position \in [0.2, 0.4] \cup [0.6, 0.8]$), and 2 to the area farthest away from the center ($position \in [0.0, 0.2] \cup [0.8, 1.0]$). This cost function was motivated by the scenario, where a human expert tries to constrain the RL agent to adhere to human advice that the agent should stay in the center. This advice is encoded as the cost function and its threshold. We experimented with various cost constraint thresholds $\hat{c} \in \{200, 100, 50, 30, 20\}$ ranging from the unconstrained case $\hat{c} = 200$ ($\because \frac{C_{\max}}{1-\gamma} = 200$) to the tightly constrained case $\hat{c} = 20$. We can see that the agent has two conflicting objectives: in order to achieve high rewards, it sometimes needs to move the paddle to positions far away from the center, but if this happens too often, the cost constraint will be violated. Thus, it needs to trade off between reward and cost properly depending on the cost constraint threshold $\hat{c}$.

Figure 2b summarizes the experimental results from the CC-POMCP and the baseline agents. When $\hat{c} = 200$ (unconstrained case), both algorithms always win the game 21 by 0. As we lower $\hat{c}$,

$|S| = 2^{1024}$, $|A| = 3$




(a) Domain description




| $\hat{c}$ | ALGO | avg cumulative rewards | avg discounted cumulative costs | avg score FOE VS ALGO |
|---|---|---|---|---|
| 200 | CC-POMCP | 21.00±0.00 | 133.00±4.97 | 0.0 vs 21.0 |
| | Baseline | 21.00±0.00 | 136.66±4.45 | 0.0 vs 21.0 |
| 100 | CC-POMCP | **19.27±1.63** | **99.88±0.13** | 1.4 vs 20.7 |
| | Baseline | −15.05±3.83 | 110.88±3.86 | 18.9 vs 3.9 |
| 50 | CC-POMCP | **17.88±1.79** | **49.95±0.07** | 2.8 vs 20.7 |
| | Baseline | −20.45±0.26 | 130.40±4.99 | 21.0 vs 0.6 |
| 30 | CC-POMCP | **-0.07±5.23** | **30.40±0.46** | 13.2 vs 13.2 |
| | Baseline | −20.48±0.30 | 131.37±5.08 | 21.0 vs 0.5 |
| 20 | CC-POMCP | −17.80±2.91 | 25.36±1.25 | 20.1 vs 2.2 |
| | Baseline | −20.48±0.30 | 131.37±5.08 | 21.0 vs 0.5 |

(b) Simulation results

Figure 2: (a) Multi-objective version of Atari 2600 PONG, visualizing the cost function. (b) Results of the constrained PONG. Above: Histogram of the CC-POMCP agent's position, where the horizontal axis denotes the position of the agent (0: topmost, 1: bottommost) and the vertical axis denotes the relative discounted visitation rate for each bin.

CC-POMCP tends to stay in the center in order to make a trade off between reward and cost (shown in the histograms in Figure 2b). We can also see that the agent gradually performs worse in terms of scores as $\hat{c}$ decreases. This is a natural result since it is forced to stay in the center and thus sacrifice the game score. Overall, CC-POMCP computes a good policy while generally respecting the cost constraint. On the other hand, the baseline fails to show a meaningful policy except when $\hat{c} = 200$ since the Monte-Carlo cost returns at early stage mostly violate the cost constraint, resulting in random behavior.

## 6 Conclusion

We presented CC-POMCP, an online MCTS algorithm for very large CPOMDPs. We showed that solving the dual LP of CPOMDPs is equivalent to jointly solving an unconstrained POMDP and optimizing its LP-induced parameters $\boldsymbol{\lambda}$, and provided theoretical results that shed insight on the properties of $\boldsymbol{\lambda}$ and how to optimize it. We then extended POMCP to maximize the scalarized value while simultaneously updating $\boldsymbol{\lambda}$ using the current action-value estimates $Q_{\mathbf{C}}$. We also empirically showed that CC-POMCP converges to the optimal stochastic actions on a toy domain and easily scales to very large CPOMDPs through the constrained variants of *Rocksample* and the multi-objective version of PONG.


## Acknowledgement

This work was supported by the ICT R&D program of MSIT/IITP of Korea (No. 2017-0-01778) and DAPA/ADD of Korea (UD170018CD). J. Lee acknowledges the Global Ph.D. Fellowship Program by NRF of Korea (NRF-2018-Global Ph.D. Fellowship Program).


## References


[1] Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[3] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[4] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:1–49, 2012.

[5] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, pages 72–83, 2006.

[6] Eugene A Feinberga and Aleksey B Piunovskiyb. Nonatomic total rewards markov decision processes with multiple criteria. *J. Math. Anal. Appl*, 273:93–111, 2002.

[7] Sylvain Gelly and David Silver. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artif. Intell.*, 175(11):1856–1875, 2011.

[8] Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-carlo tree search. *Journal of Artificial Intelligence Research*, pages 841–883, 2013.

[9] Joshua D. Isom, Sean P. Meyn, and Richard D. Braatz. Piecewise linear dynamic programming for constrained POMDPs. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 291–296, 2008.

[10] Sammie Katt, Frans A. Oliehoek, and Christopher Amato. Learning in POMDPs with Monte Carlo tree search. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1819–1827, 2017.

[11] Dongho Kim, Jaesong Lee, Kee-Eung Kim, and Pascal Poupart. Point-based value iteration for constrained POMDPs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 1968–1974, 2011.

[12] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning (ECML 2006)*, pages 282–293, 2006.

[13] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved Monte-Carlo Search. Technical Report 1, Univ. Tartu, Estonia, 2006.

[14] Jongmin Lee, Youngsoo Jang, Pascal Poupart, and Kee-Eung Kim. Constrained Bayesian reinforcement learning via approximate linear programming. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 2088–2095, 2017.

[15] Eunsoo Oh and Kee-Eung Kim. A geometric traversal algorithm for reward-uncertain MDPs. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 565–572, 2011.

[16] Christos Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

[17] Alexei B Piunovskiy and Xuerong Mao. Constrained Markovian decision processes: the dynamic programming approach. *Operations research letters*, 27(3):119–126, 2000.

[18] Pascal Poupart, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling. Approximate linear programming for constrained partially observable Markov decision processes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3342–3348, 2015.

[19] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems 23*, pages 2164–2172, 2010.

[20] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, pages 484–489, 2016.

[21] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 520–527, 2004.

[22] Edward J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

[23] A. Undurti and J. P. How. An online algorithm for constrained POMDPs. In *2010 IEEE International Conference on Robotics and Automation*, pages 3966–3973, 2010.

[24] Jason D. Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422, 2007.

# Supplementary Material: Monte-Carlo Tree Search for Constrained POMDPs (Jongmin Lee, Geon-Hyeong Kim, Pascal Poupart, Kee-Eung Kim)

## Appendix A Proof of Proposition 1

**Proposition 1.** *Let $\mathcal{V}^*_{\boldsymbol{\lambda}}$ be the optimal value function of the POMDP with scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. Then, $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}}$ is a piecewise-linear and convex (PWLC) function over $\boldsymbol{\lambda}$.*

*Proof.* We give a proof by induction. For all $b$,

$$\mathcal{V}^{(0)}_{\boldsymbol{\lambda}}(b) = \max_a \Big[ R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a) \Big]$$

is a piecewise-linear and convex function over $\boldsymbol{\lambda}$ since the max of linear functions is piecewise linear and convex. Now, assume the following induction hypothesis: $\mathcal{V}^{(k)}_{\boldsymbol{\lambda}}(b)$ is piecewise-linear and convex function over $\boldsymbol{\lambda}$ for all $b$. Then, for all $b$,

$$\mathcal{V}^{(k+1)}_{\boldsymbol{\lambda}}(b) = \max_a \Big[ \underbrace{R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)}_{\text{linear in } \boldsymbol{\lambda}} + \gamma \sum_{o,s,s'} \underbrace{Z_p(o|s',a) T_p(s'|s,a) b(s) \mathcal{V}^{(k)}_{\boldsymbol{\lambda}}(b^{ao})}_{\text{PWLC in } \boldsymbol{\lambda}} \Big]$$

is also PWLC since the summation of PWLC functions is PWLC and max over PWLC functions is again PWLC. As a consequence, $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) = \lim_{k\to\infty} \mathcal{V}^{(k)}_{\boldsymbol{\lambda}}(b_0)$ is PWLC over $\boldsymbol{\lambda}$ and so is $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}}$. $\square$

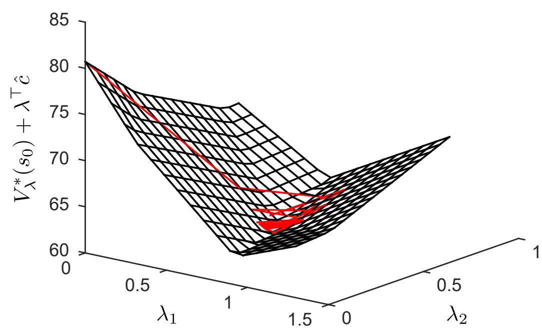



Figure 3: $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}}$ for a simple CPOMDP, which is piecewise-linear and convex. Red line represents the trajectory of $\boldsymbol{\lambda}$ starting from $\boldsymbol{\lambda} = [0,0]^\top$ and sequentially updated by $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha_n (V^{\pi^*_{\boldsymbol{\lambda}}}_{\mathbf{C}}(b_0) - \hat{\mathbf{c}})$.

## Appendix B Proof of Lemma 1

**Lemma 1.** *Let $\mathcal{M}_1 = \langle B, A, T, R_1, \gamma \rangle$ and $\mathcal{M}_2 = \langle B, A, T, R_2, \gamma \rangle$ be two (belief-state) MDPs differing only in the reward function, and $V_1^\pi$ and $V_2^\pi$ be the value functions of $\mathcal{M}_1$ and $\mathcal{M}_2$ with a fixed policy $\pi$. Then, the value function of the new MDP $\mathcal{M} = \langle B, A, T, pR_1 + qR_2, \gamma \rangle$ with the policy $\pi$ is $V^\pi(b) = pV_1^\pi(b) + qV_2^\pi(b)$ for all $b \in B$.*

*Proof.* We give a proof by induction. For all $b$,

$$\begin{aligned} V^{(0)}(b) &= \sum_a \pi(a|b) \left[ pR_1(b,a) + qR_2(b,a) \right] \\ &= p \sum_a \pi(a|b) R_1(b,a) + q \sum_a \pi(a|b) R_2(b,a) \\ &= pV_1^{(0)}(b) + qV_2^{(0)}(b) \end{aligned}$$

Then, assume the induction hypothesis $V^{(k)}(b) = pV_1^{(k)}(b) + qV_2^{(k)}(b)$. For all $b$,

$$
\begin{aligned}
&V^{(k+1)}(b)\\
&= \sum_a \pi(a|b)\left[pR_1(b,a) + qR_2(b,a) + \gamma \sum_{b'} T(b'|b,a)V^{(k)}(b')\right]\\
&= p\sum_a \pi(a|b)\left[R_1(b,a) + \gamma \sum_{b'} T(b'|b,a)V_1^{(k)}(b')\right]\\
&\quad + q\sum_a \pi(a|b)\left[R_2(b,a) + \gamma \sum_{b'} T(b'|b,a)V_2^{(k)}(b')\right]\\
&= pV_1^{(k+1)}(b) + qV_2^{(k+1)}(b)
\end{aligned}
$$

As a consequence, $V^{\pi}(b) = \lim_{k\to\infty} V^{(k)}(b) = \lim_{k\to\infty}(V_1^{(k)}(b) + V_2^{(k)}(b)) = V_1^{\pi}(b) + V_2^{\pi}(b)$. □

# Appendix C Proof of Theorem 1

**Theorem 1.** *For any $\boldsymbol{\lambda}$, $V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}}}(b_0) - \hat{\mathbf{c}}$ is a negative subgradient that decreases the objective in Eq.* (3)*, where $\pi^*_{\boldsymbol{\lambda}}$ is the optimal policy with respect to the scalarized reward function $R(b,a) - \boldsymbol{\lambda}^\top \mathbf{C}(b,a)$. Also, if $V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}}}(b_0) - \hat{\mathbf{c}} = \mathbf{0}$ then $\boldsymbol{\lambda}$ is the optimal solution of Eq.* (3)*.*

*Proof.* For any $\boldsymbol{\lambda}_0$ and $\boldsymbol{\lambda}_1$,

$$
\begin{aligned}
&(\mathcal{V}^*_{\boldsymbol{\lambda}_1}(b_0) + \boldsymbol{\lambda}_1^\top \hat{\mathbf{c}}) - (\mathcal{V}^*_{\boldsymbol{\lambda}_0}(b_0) + \boldsymbol{\lambda}_0^\top \hat{\mathbf{c}})\\
=&(V_R^{\pi^*_{\boldsymbol{\lambda}_1}}(b_0) - \boldsymbol{\lambda}_1^\top V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_1}}(b_0)) - (V_R^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0) - \boldsymbol{\lambda}_0^\top V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0)) + (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0)^\top \hat{\mathbf{c}} && \text{(Lemma 1)}\\
\ge&(V_R^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0) - \boldsymbol{\lambda}_1^\top V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0)) - (V_R^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0) - \boldsymbol{\lambda}_0^\top V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0)) + (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0)^\top \hat{\mathbf{c}}\\
& && (\because \pi^*_{\boldsymbol{\lambda}_1} \text{ is optimal w.r.t. } R - \boldsymbol{\lambda}_1^\top \mathbf{C})\\
=&(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0)^\top(\hat{\mathbf{c}} - V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0)) && (11)
\end{aligned}
$$

Therefore, $\hat{\mathbf{c}} - V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0)$ is a *subgradient* of $\mathcal{V}^*_{\boldsymbol{\lambda}}(b_0) + \boldsymbol{\lambda}^\top \hat{\mathbf{c}}$ at point $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$, which concludes that $V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0) - \hat{\mathbf{c}}$ is a *negative subgradient* at point $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$.

In addition, suppose $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_0 + \alpha(V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0) - \hat{\mathbf{c}})$. Then, for sufficiently small $\alpha > 0$, the new reward function $R - \boldsymbol{\lambda}_1^\top \mathbf{C}$ which is slightly changed from the old reward $R - \boldsymbol{\lambda}_0^\top \mathbf{C}$ still satisfies the *reward optimality condition* with respect to policy $\pi^*_{\boldsymbol{\lambda}_0}$ [15]. In this situation, $\pi^*_{\boldsymbol{\lambda}_0} = \pi^*_{\boldsymbol{\lambda}_1}$ and we obtain:

$$
\begin{aligned}
(\mathcal{V}^*_{\boldsymbol{\lambda}_0}(b_0) + \boldsymbol{\lambda}_0^\top \hat{\mathbf{c}}) - (\mathcal{V}^*_{\boldsymbol{\lambda}_1}(b_0) + \boldsymbol{\lambda}_1^\top \hat{\mathbf{c}}) &\ge (\boldsymbol{\lambda}_0 - \boldsymbol{\lambda}_1)^\top(\hat{\mathbf{c}} - V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_1}}(b_0)) && \big(\text{by result of (11)}\big)\\
&= -\alpha(V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_0}}(b_0) - \hat{\mathbf{c}})^\top(\hat{\mathbf{c}} - V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_1}}(b_0))\\
&= \alpha\|\hat{\mathbf{c}} - V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}_1}}(b_0))\|_2^2\\
&\ge 0
\end{aligned}
$$

Also, if $V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}}}(b_0) - \hat{\mathbf{c}} = \mathbf{0}$, then the dual objective in Eq. (4) becomes $V_R^{\pi^*_{\boldsymbol{\lambda}}}(b_0)$. By the weak duality theorem, $V_R^{\pi^*_{\boldsymbol{\lambda}}}(b_0) \ge \sum_{b,a} R(b,a)y^*(b,a) = V_R^*(b_0;\hat{\mathbf{c}})$ holds where $y^*(b,a)$ is the optimal solution of the primal LP (1). Assuming that $V_{\mathbf{C}}^{\pi^*_{\boldsymbol{\lambda}}}(b_0) = \hat{\mathbf{c}}$, $\pi^*_{\boldsymbol{\lambda}}$ is a feasible policy that satisfies the cost constraints, and $V_R^{\pi^*_{\boldsymbol{\lambda}}}(b_0)$ cannot be larger than $V_R^*(b_0;\hat{\mathbf{c}})$. That is, $V_R^{\pi^*_{\boldsymbol{\lambda}}}(b_0) = V_R^*(b_0;\hat{\mathbf{c}})$ and the duality gap is zero, which means that $\boldsymbol{\lambda}$ is the optimal solution. □

## Appendix D Recursive Update of Admissible Cost

As an alternative to using $V_C^*(h_{t+1})$, we can also do recursive update of the admissible cost at time step $t+1$ to depend only on the current expected immediate cost at $(h_t, a_t)$ and the current optimal cost value $Q_{\mathbf{C}}^{\pi^*}(h_t, a_t)$ by the following relationship.

$$
\begin{aligned}
\hat{\mathbf{c}}_t &= V_{\mathbf{C}}^{\pi^*}(h_t) = \sum_a \pi^*(a|h_t) Q_{\mathbf{C}}^{\pi^*}(h_t, a) \\
&= \pi^*(a_t|h_t) Q_{\mathbf{C}}^{\pi^*}(h_t, a_t) + \sum_{a \neq a_t} \pi^*(a|h_t) Q_{\mathbf{C}}^{\pi^*}(h_t, a) \\
&= \pi^*(a_t|h_t) \Big[ \mathbf{C}(h_t, a_t) + \gamma \underbrace{\mathbb{E}[V_{\mathbf{C}}^{\pi^*}(h_{t+1}) | h_t, a_t]}_{= \hat{\mathbf{c}}_{t+1}} \Big] + \sum_{a \neq a_t} \pi^*(a|h_t) Q_{\mathbf{C}}^{\pi^*}(h_t, a)
\end{aligned}
$$

$$
\therefore \; \hat{\mathbf{c}}_{t+1} = \frac{\hat{\mathbf{c}}_t - \pi^*(a_t|h_t)\mathbf{C}(h_t, a_t) - \sum_{a \neq a_t} \pi^*(a|h_t) Q_{\mathbf{C}}^{\pi^*}(h_t, a)}{\gamma \pi(a_t|b_t)} \tag{12}
$$

We used Eq. (12) to update the admissible cost in the experiments.

## Appendix E Equality Test for Collecting Optimal Action Candidates

**Algorithm 2** SEARCH of CC-POMCP

**function** SEARCH($h_0$)
  $\boldsymbol{\lambda}$ is randomly initialized
  **for** $n = 1, 2, \ldots$ **do**
    **for** $t = 1, 2, \ldots, f(n)$ **do**  # $f(n)$ is any monotonically increasing sequence w.r.t. $n$.
      **if** $h = \emptyset$ **then**
        $s \sim b_0$
      **else**
        $s \sim \mathcal{B}(h_0)$
      **end if**
      SIMULATE($s, h_0, 0$)
    **end for**
    $a \sim$ GREEDYPOLICY($h_0, 0, 0$)
    $\lambda \leftarrow \lambda + \alpha_n [Q_C(h_0, a) - \hat{c}]$
    Clip $\lambda_k$ to range $[0, \frac{R_{\max} - R_{\min}}{\tau(1-\gamma)}]$  $\forall k = \{1, 2, \ldots K\}$
    Reset entire search tree $T(h_0)$
  **end for**
  **return** GREEDYPOLICY($h_0, 0, \nu$)
**end function**

To compute a stochastic optimal policy, we first need to collect the candidates of the support of the optimal actions. Since there always exists some error for estimating $\mathcal{Q}_{\boldsymbol{\lambda}}(h, a)$ due to the randomness of Monte-Carlo sampling, we need an explicit criterion to determine whether an action $\hat{a}$ can be treated as an optimum (i.e. $\mathcal{Q}_{\boldsymbol{\lambda}}(h, \hat{a}) \simeq Q_{\boldsymbol{\lambda}}(h, a^*)$). The following theorem provides the *equality test* criterion to collect optimal action candidates and its validity.

**Theorem 3.** *Let $\mathcal{Q}_{\boldsymbol{\lambda}}$ be the scalarized action-value estimated by CC-POMCP using* SEARCH *in Algorithm 2, and $a^* = \arg\max_a \mathcal{Q}_{\boldsymbol{\lambda}}^{\oplus}(h, a)$. For any $\nu > 0$, suppose we use the following equality test criterion for the optimal action selection: $|\mathcal{Q}_{\boldsymbol{\lambda}}(h, a^*) - \mathcal{Q}_{\boldsymbol{\lambda}}(h, \hat{a})| \leq \nu \left[ \sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} + \sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}} \right]$. Then it accepts all of the optimal actions (with respect to MDP with the scalarized reward function $R(s,a) - \boldsymbol{\lambda}^\top \mathbf{C}(s,a)$) while rejecting all of the suboptimal actions with probability 1 as $t \to \infty$.*

To prove Theorem 3, we first introduce a lemma from [13]:

**Lemma 2** (Theorem 7 in [13]). *When the UCT algorithm is running on a tree with depth $D$, the bias of expected payoff is $O((|A| D \log t + |A|^D)/t)$ after $t$ iteration. Moreover, the probability that UCT algorithm fails to select optimal action at root converges to zero as $t \to \infty$.*

By Lemma 2, there exists $M > 0$ such that the following holds:

$$\Pr\left(|\mathcal{Q}^*_{\boldsymbol{\lambda}}(h,a) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,a)| \geq M\frac{\log N(h,a)}{N(h,a)}\right) \to 0 \text{ as } N(h,a) \to \infty, \tag{13}$$

where $\mathcal{Q}^*_{\boldsymbol{\lambda}} = \mathcal{Q}^{\pi^*_{\boldsymbol{\lambda}}}_{\boldsymbol{\lambda}}$ (true optimal value function of MDP with the scalarized reward function). In order to prove Theorem 3, we first provide the following two lemmas based on Eq. (13).

**Lemma 3.** *Let $a^* = \arg\max_a \mathcal{Q}^{\oplus}_{\boldsymbol{\lambda}}(h,a)$, and $\hat{a}$ be a suboptimal action (with respect to MDP with the scalarized reward function $R(s,a) - \boldsymbol{\lambda}^\top \mathbf{C}(s,a)$). For the given $\boldsymbol{\lambda}$, if $a^*$ is an optimal action, then $\Pr\left(|\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a})| \leq \nu\left[\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} + \sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}}\right]\right) \to 0$ as $t \to \infty$. In other words, all of the suboptimal actions are rejected asymptotically by the proposed equality test criterion.*

*Proof.* Let $\mathcal{Q}^*_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}^*_{\boldsymbol{\lambda}}(h,\hat{a}) = \Delta > 0$, where $\mathcal{Q}^*_{\boldsymbol{\lambda}} = \mathcal{Q}^{\pi^*_{\boldsymbol{\lambda}}}_{\boldsymbol{\lambda}}$. Then using the fact that $\Pr(A+B < C+D) \leq \Pr(A < C) + \Pr(B < D)$, we can derive the following inequality:

$$\begin{aligned}
&\Pr\left(\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a}) \leq \nu\left[\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} + \sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}}\right]\right) \\
&\leq \Pr\left(\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}^*_{\boldsymbol{\lambda}}(h,a^*) + \frac{\Delta}{2} \leq \nu\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}}\right) \\
&\quad + \Pr\left(\mathcal{Q}^*_{\boldsymbol{\lambda}}(h,\hat{a}) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a}) + \frac{\Delta}{2} \leq \nu\sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}}\right).
\end{aligned} \tag{14}$$

Therefore, it is enough to show that both of two terms in the right-hand side of (14) converge to 0 as $t \to \infty$. It is obvious that the first term converges to probability 0 since $\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}^*_{\boldsymbol{\lambda}}(h,a^*) \to 0$ by Eq. (13) and $\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} \to 0$ as $N(h,a^*) \to \infty$. The second term also converges to probability 0 by the same reasoning. □

**Lemma 4.** *Let $a^* = \arg\max_a \mathcal{Q}^{\oplus}_{\boldsymbol{\lambda}}(h,a)$, and $\hat{a}$ be another optimal action (with respect to MDP with the scalarized reward function $R(s,a) - \boldsymbol{\lambda}^\top \mathbf{C}(s,a)$). For the given $\boldsymbol{\lambda}$, if $a^*$ is an optimal action, then $\Pr\left(|\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a})| > \nu\left[\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} + \sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}}\right]\right) \to 0$ as $t \to \infty$. In other words, all of the optimal actions are accepted asymptotically by the proposed equality test criterion.*

*Proof.* Without loss of generality, assume that $\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) \geq \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a})$ at time $t$. Then, we can derive the following inequality, similarly to Lemma 3:

$$\begin{aligned}
&\Pr\left(\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a}) > \nu\left[\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} + \sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}}\right]\right) \\
&\leq \Pr\left(\mathcal{Q}_{\boldsymbol{\lambda}}(h,a^*) - \mathcal{Q}^*_{\boldsymbol{\lambda}}(h,a^*) > \nu\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}}\right) \\
&\quad + \Pr\left(\mathcal{Q}^*_{\boldsymbol{\lambda}}(h,\hat{a}) - \mathcal{Q}_{\boldsymbol{\lambda}}(h,\hat{a}) > \nu\sqrt{\frac{\log N(h,\hat{a})}{N(h,\hat{a})}}\right)
\end{aligned} \tag{15}$$

where $\mathcal{Q}^*_{\boldsymbol{\lambda}} = \mathcal{Q}^{\pi^*_{\boldsymbol{\lambda}}}_{\boldsymbol{\lambda}}$. Note that $\mathcal{Q}^*_{\boldsymbol{\lambda}}(h,a^*) = \mathcal{Q}^*_{\boldsymbol{\lambda}}(h,\hat{a})$ since the both actions $a^*$ and $\hat{a}$ are optimal. It is sufficient to show that both of two terms in the right-hand side of Eq. (15) converge to 0 as $t \to \infty$. For any $M > 0$, the inequality

$$\nu\sqrt{\frac{\log N(h,a^*)}{N(h,a^*)}} \geq M\frac{\log N(h,a^*)}{N(h,a^*)}$$

holds as $N(h,a^*) \to \infty$. Therefore, by Eq. (13), the first term converges to probability 0. The second term also converges to probability 0 by the same reasoning. □

Finally, we are now ready to provide the proof of Theorem 3:

*Proof of Theorem 3.* Let $B$ the event that the proposed equality test accepts all of the optimal actions while rejecting all of the suboptimal actions. Let $a^* = \arg\max_a \mathcal{Q}^{\oplus}_{\boldsymbol{\lambda}}(h, a)$. Then, for $B$ not to be satisfied, one of the following three cases should be hold:

1. $B_1$: $a^*$ is not optimal action (i.e. $a^* \notin \arg\max Q^{\pi^*_{\boldsymbol{\lambda}}}_{\boldsymbol{\lambda}}(h, a)$) with respect to the scalarized reward $R - \boldsymbol{\lambda}^\top \mathbf{C}$.
2. $B_2$: $\hat{a}$ is a suboptimal action but it is accepted as an equally optimal action while $a^*$ is an optimal action.
3. $B_3$: $\hat{a}$ is an optimal action but it is rejected while $a^*$ is an optimal action.

Then,

$$\Pr(B) \geq 1 - \Pr(B_1) - \Pr(\neg B_1 \cap B_2) - \Pr(\neg B_1 \cap B_3).$$

Here, $\Pr(B_1)$, $\Pr(B_2)$, and $\Pr(B_3)$ converge to zero as $t \to \infty$ by Lemma 2, 3, and 4 respectively, which concludes the proof. □

## Appendix F Proof of Theorem 2

To facilitate formal analysis, we assume that SEARCH of CC-POMCP is given by Algorithm 2. We first introduce the following lemma.

**Lemma 5.** *For any real numbers $V^*, V$, and $c > 0$, if there is $M > 0$ such that $|V^* - V| \leq M$ and $|V^* - c| > M$, then $(V^* - c)(V - c) > 0$.*

*Proof.*

$$\begin{aligned}(V^* - c)(V - c) &= (V^* - c)^2 + (V^* - c)(V - V^*) \\ &\geq (V^* - c)^2 - |V^* - c| \cdot |V - V^*| \\ &= |V^* - c|\big(|V^* - c| - |V - V^*|\big) \\ &> |V^* - c|\,(M - M) \\ &= 0.\end{aligned}$$

□

Now, we are ready to provide the proof of the following theorem, which guarantees that $\boldsymbol{\lambda}$ is improved until it converges to the optimal solution of Eq. (3), $\boldsymbol{\lambda}^*$.

**Theorem 2.** *Suppose $\boldsymbol{\lambda}$ is updated with increasing simulation step $t$, and the search tree is reset at the end of $\boldsymbol{\lambda}$'s update as outlined in Algorithm 2. If the asymptotic bias of UCT holds for all types of cost values (i.e. $\exists M > 0, \forall k$, $|V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - V_{C_k}(h_0)| \leq M\frac{\log t}{t}$), then either $sign(V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - \hat{c}_k) = sign(V_{C_k}(h_0) - \hat{c}_k)$ or $|V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - \hat{c}_k| \leq M\frac{\log t}{t}$ holds with probability 1 as $t \to \infty$, where $V_{C_k}(h_0)$ is an averaged Monte-Carlo return for $k$-th cost at the root node $h_0$.*

*Proof.* If $|V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - \hat{c}_k| \leq M\frac{\log t}{t}$, the statement trivially holds. If $|V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - \hat{c}_k| > M\frac{\log t}{t}$, then by Lemma 5 and the assumption $|V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - V_{C_k}(h_0)| \leq M\frac{\log t}{t}$,

$$(V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - \hat{c}_k)(V_{C_k}(h_0) - \hat{c}_k) > 0$$

which concludes $sign(V^{\pi^*_{\boldsymbol{\lambda}}}_{C_k}(h_0) - \hat{c}_k) = sign(V_{C_k}(h_0) - \hat{c}_k)$. □

Even though Theorem 2 asymptotically guarantees that $\boldsymbol{\lambda}$ is near-optimal or it is updated by the direction of the negative subgradient, it requires *resetting* the entire search tree as described in Algorithm 2, which significantly degrades the sample efficiency of the algorithm. Fortunately, we can further show that the policy ordering is locally preserved even when $\boldsymbol{\lambda}$ is changed slightly, which justifies the use of practical Algorithm 1 that does not reset the tree and accumulates experiences on different lambda into single search tree.

**Theorem 4.** *For any $\boldsymbol{\lambda}_0$, there exists $\epsilon > 0$ such that if $\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0\|_1 < \epsilon$ then $\forall b, \mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) > \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b)$ $\Rightarrow \forall b, \mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_1}(b) > \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_1}(b)$. In other words, for a sufficiently small change of $\boldsymbol{\lambda}$, the ordering of the policies is preserved.*

*Proof.* Let $\Delta\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0$. Then, for any $b$ and $\pi$,

$$\begin{aligned}
\mathcal{V}^{\pi}_{\boldsymbol{\lambda}_1}(b) &= V^{\pi}_R(b) - \boldsymbol{\lambda}_1^\top V^{\pi}_{\mathbf{C}}(b) && \text{(Lemma 1)} \\
&= V^{\pi}_R(b) - (\boldsymbol{\lambda}_0 + \Delta\boldsymbol{\lambda})^\top V^{\pi}_{\mathbf{C}}(b) \\
&= \mathcal{V}^{\pi}_{\boldsymbol{\lambda}_0}(b) - \Delta\boldsymbol{\lambda}^\top V^{\pi}_{\mathbf{C}}(b)
\end{aligned}$$

Now assume $\forall b$, $\mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) > \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b)$, let $\epsilon = \min_b \left[\frac{1-\gamma}{C_{\max}}(\mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b))\right] > 0$, and suppose $\|\Delta\boldsymbol{\lambda}\|_1 = \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0\|_1 < \epsilon$. Then, for any $b$,

$$\begin{aligned}
\mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_1}(b) - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_1}(b) &= \mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b) - \Delta\boldsymbol{\lambda}^\top (V^{\pi_1}_{\mathbf{C}}(b) - V^{\pi_0}_{\mathbf{C}}(b)) \\
&\geq \mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b) - \|\Delta\boldsymbol{\lambda}\|_1 \|V^{\pi_1}_{\mathbf{C}}(b) - V^{\pi_0}_{\mathbf{C}}(b)\|_\infty && \text{(Hölder's inequality)} \\
&> \mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b) - \epsilon \frac{C_{\max}}{1-\gamma} \\
&= \mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b) - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b) - \min_{b'} \left[(\mathcal{V}^{\pi_1}_{\boldsymbol{\lambda}_0}(b') - \mathcal{V}^{\pi_0}_{\boldsymbol{\lambda}_0}(b'))\right] \geq 0
\end{aligned}$$

□

## Appendix G Analytic Solution of LP (10) (when $K = 1$)

When $K = 1$ and $\lambda > 0$, LP (10) can be rewritten as:

$$\begin{aligned}
&\min_{\{w_i, \xi^+, \xi^-\}} (\xi^+ + \xi^-) \\
&s.t. \sum_{i: a_i^* \in A^*} w_i Q_C(h, a_i^*) = \hat{c} + (\xi^+ - \xi^-) \\
&\quad\;\; \sum_{i: a_i^* \in A^*} w_i = 1 \;\text{ and }\; w_i, \xi^+, \xi^- \geq 0
\end{aligned}$$

Let $a_{\min} = \arg\min_{a_i^* \in A^*} Q_C(h, a_i^*)$ and $a_{\max} = \arg\max_{a_i^* \in A^*} Q_C(h, a_i^*)$. Then, the analytic solution is given by:

$$\pi(a_{\min}|h) = \begin{cases} 0 & \text{if } Q_C(h, a_{\max}) \leq \hat{c} \\ 1 & \text{if } Q_C(h, a_{\min}) \geq \hat{c} \\ \frac{Q_C(h, a_{\max}) - \hat{c}}{Q_C(h, a_{\max}) - Q_C(h, a_{\min})} & \text{if } Q_C(h, a_{\min}) < \hat{c} < Q_C(h, a_{\max}) \end{cases}$$

and

$$\pi(a_{\max}|h) = \begin{cases} 1 & \text{if } Q_C(h, a_{\max}) \leq \hat{c} \\ 0 & \text{if } Q_C(h, a_{\min}) \geq \hat{c} \\ \frac{\hat{c} - Q_C(h, a_{\min})}{Q_C(h, a_{\max}) - Q_C(h, a_{\min})} & \text{if } Q_C(h, a_{\min}) < \hat{c} < Q_C(h, a_{\max}) \end{cases}$$

This has only $O(|A|)$ time complexity, which is identical to that of UCB1 action selection.

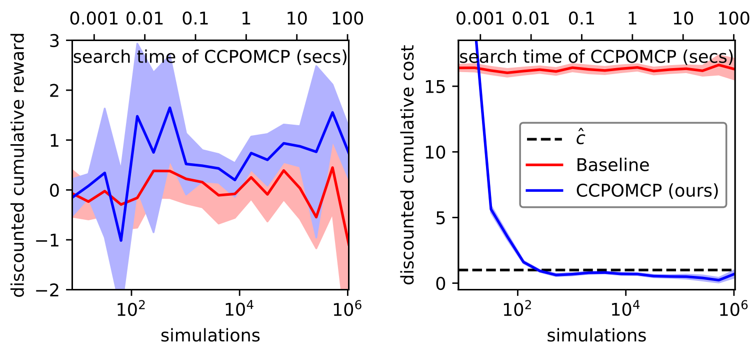
## Appendix H Experimental Setup

We used the following experimental parameters for each domain.

| Domain | **Toy** | **Rocksample** | **Atari Pong** |
|---|---|---|---|
| $\kappa$ | 1 | 20 | 0.1 |
| $\tau$ | $\hat{c} = 0.95$ | $\hat{c} = 1$ | $\hat{c} \in \{20, 30, 50, 100, 200\}$ |
| $\alpha_n$ | $1/n$ | $1/n$ | $1/n$ |
| $\nu$ | 1 | 1 | 1 |
| # of runs | 1000 | 100 (*) | 40 |
| # of simulations | from $2^3$ to $2^{20}$ | from $2^3$ to $2^{20}$ | 1000 |
| maximum-depth $d$ | $\gamma^d = 0.001$ | $\gamma^d = 0.001$ | $d = 100$ |

Table 2: (*) We report the result averaged over 100 runs or 12 hours of total computation time.

# Appendix I Experimental results on Rocksample (15, 15)




Note that the baseline agent (red) is violating the cost constraint, so its return is not meaningful at all.