Bounded-Error Policy Optimization for Mixed Discrete-Continuous MDPs via Constraint Generation in Nonlinear Programming

Michael Gimelfarb[1][0000−0003−4377−2142], Ayal Taitler[2][0000−0003−3919−6883], and Scott Sanner[3][0000−0001−7984−8394]

[1] Department of Computer Science, University of Toronto, Canada
mike.gimelfarb@mail.utoronto.ca
[2] Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Be'er Sheva, Israel
ataitler@bgu.ac.il
[3] Department of Mechanical and Industrial Engineering, University of Toronto, Canada
ssanner@mie.utoronto.ca

Abstract. We propose the Constraint-Generation Policy Optimization (CGPO) framework to optimize policy parameters within compact and interpretable policy classes for mixed discrete-continuous Markov Decision Processes (DC-MDP). CGPO can not only provide bounded policy error guarantees over an infinite range of initial states for many DC-MDPs with expressive nonlinear dynamics, but it can also provably derive optimal policies in cases where it terminates with zero error. Furthermore, CGPO can generate worst-case state trajectories to diagnose policy deficiencies and provide counterfactual explanations of optimal actions. To achieve such results, CGPO proposes a bilevel mixed-integer nonlinear optimization framework for optimizing policies in defined expressivity classes (e.g. piecewise linear) and reduces it to an optimal constraint generation methodology that adversarially generates worst-case state trajectories. Furthermore, leveraging modern nonlinear optimizers, CGPO can obtain solutions with bounded optimality gap guarantees. We handle stochastic transitions through chance constraints, providing high-probability performance guarantees. We also present a roadmap for understanding the computational complexities of different expressivity classes of policy, reward, and transition dynamics. We experimentally demonstrate the applicability of CGPO across various domains, including inventory control, management of a water reservoir system, and physics control. In summary, CGPO provides structured, compact and explainable policies with bounded performance guarantees, enabling worst-case scenario generation and counterfactual policy diagnostics.

Keywords: Planning · Control · Mixed Discrete-Continuous MDP · Constraint Generation · Chance Constraints · Piecewise-Linear Policy · Sequential Decision Optimization · Policy Optimization

1 Introduction

An important aim of sequential decision optimization of challenging *Discrete-Continuous Markov Decision Processes* (DC-MDP) in the artificial intelligence, operations research, and control domains is to derive policies that achieve optimal control. A desirable property of such policies is *compactness* of representation, which provides efficient execution on resource-constrained systems such as mobile devices [41], as well as the potential for *introspection and explanation* [39]. Moreover, while the derived policy is expected to perform well in expectation, in many applications it is desirable to obtain *bounds on maximum policy error* and the scenarios that induce worst-case policy performance [13].

Popular policy optimization approaches used in model-free reinforcement learning [35] do not provide bounded error guarantees on policy performance, even if they use compact policy classes such as decision trees [39]. Model-based extensions that leverage gradient-based policy optimization [10] similarly do not provide bounded error guarantees due to the nonconvexity of the problem.

In contrast, there exists a rich tradition of work leveraging *mixed integer programming* (MIP) for bounded-error policy optimization [2,14,40], but these methods only apply to discrete state and action MDPs. More importantly, these previous policy optimization formulations cannot be directly extended to continuous state or action MDPs due to the infinite number of constraints arising from the infinite state and action space in these formulations. While some historical work has attempted to circumvent large or infinite constraint sets (from the continuous setting) via constraint generation [15,18,36], none of these works optimize for policies. Finally, while there exist bespoke bounded error policy solutions for niche MDP classes with continuous states or actions such as linear–quadratic controllers [5], ambulance dispatching [3], and the celebrated "(s, S)" threshold policies for [34]'s inventory management, these solution methods do not readily generalize beyond the specialized domains for which they are designed.

It remains an open question as to how to derive bounded error policies for the infinite state and action, discrete and continuous MDP (DC-MDP) setting. We address this open question through a novel bilevel MIP formulation. Unfortunately, unlike MIP solutions for discrete MDPs, DC-MDPs with continuous states and/or actions cannot be solved directly since the resulting MIP (a) requires a challenging bilevel MIP formulation, (b) has an uncountably infinite number of constraints, and (c) will often be nonlinear. Fortuitously, we show how a constraint generation form of this policy optimization – termed CGPO – is able to address (a), (b) and (c) to optimize policies with bounded error guarantees in many cases.

In summary, our aim is to provide a solution approach to optimize and provide strong performance bound guarantees on structured and compact DC-MDP policies under various expressivity classes of policy, reward, and nonlinear transition dynamics. Our specific contributions are:

1. *We propose a novel bilevel optimization framework for solving nonlinear DC-MDPs called Constraint-Generation Policy Optimization (CGPO), that admits a clever reduction to an iterative constraint-generation algorithm that*

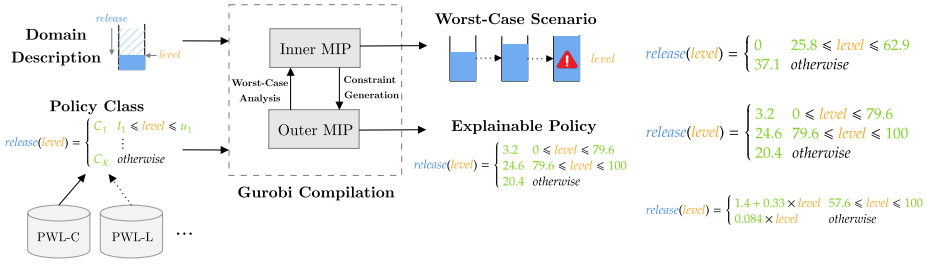


Fig. 1. Reservoir control is used as an illustrative example. **Left:** an overview of CGPO, which consists of a domain description and policy representation compiled to a bilevel mixed-integer program (MIP), in which the inner problem computes the worst-case trajectories for the current policy while the outer problem updates the policy via constraint-generation. The result is a worst-case scenario for the policy (facilitating policy failure analysis), a concrete policy within the expressivity class (for direct policy inspection), and a gap on its performance (error bound). **Right:** three optimal (i.e. zero-gap) policies produced upon termination across several piecewise policy classes. Crucially, our framework provides the ability to derive highly compact (e.g. memory and time-efficient to execute), intuitive and nonlinear policies, with strong bound guarantees on policy performance.

can be readily implemented using standard MIP solvers (Fig. 1). State-of-the-art MIP solvers often leverage spatial branch-and-bound techniques, which can provide not only optimal solutions for large mixed integer linear programs (MILP), but also bounded optimality guarantees for mixed integer nonlinear programs (MINLP) [11]. Chance constraints [16] are used for probabilistic guarantees.

2. *If the constraint generation algorithm terminates, we guarantee that we have found the optimal policy within the given policy expressivity class (Theorem 1).* Further, even when constraint generation does not terminate, our algorithm provides a tight optimality bound on the performance of the computed policy at each iteration and the scenario where the policy performs worst (and as a corollary, for any externally provided policy).
3. *We provide a road map to characterize the optimization problems in (1) above – and their associated expressivity classes – for different expressivity classes of policies and state dynamics, ranging from MILP, to polynomial programming, to nonlinear programs (Table 1).* This information is beneficial for reasoning about which optimization techniques are most effective for different combinations of DC-MDP and policy class expressivity.
4. Finally, *we provide a variety of experiments demonstrating the range of rich applications of CGPO under linear and nonlinear dynamics and various policy classes.* Critically, we derive bounded optimal solutions for a range of problems, including linear inventory management, reservoir control, and a highly nonlinear VTOL control problem. Furthermore, since our policy classes are compact by design, we can also directly inspect and analyze these

policies (Fig. 4). To facilitate policy interpretation and diagnostics, we can compute the state and exogenous noise trajectory scenario that attains the worst-case error bounds for a policy (Fig. 7) as well as a counterfactual explanation of what action should have been taken in comparison to what the policy prescribes.

2 Preliminaries

2.1 Function Classes

We first define some important classes of functions commonly referred to in the paper. Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary sets. A function $f : \mathcal{X} \to \mathcal{Y}$ is called *piecewise* (PW) if there exist functions $f_1, \dots f_K, f_{K+1} : \mathcal{X} \to \mathcal{Y}$ and disjoint Boolean case conditions $\mathcal{P}_1, \dots \mathcal{P}_K : \mathcal{X} \to \{0, 1\}$, such that for any $x \in \mathcal{X}$, $\forall i, j \neq i$ we have $\mathcal{P}_i(x) = 1 \Rightarrow \mathcal{P}_j(x) = 0$, and for case functions $f(x) = f_i(x)$ if $\mathcal{P}_i(x) = 1$ and $f(x) = f_{K+1}(x)$ if $\mathcal{P}_i(x) = 0$, $\forall i$.

Both case conditions $\mathcal{P}_i$ and case functions f_i can be linear or nonlinear. In this paper, we consider the following classes:

C *constant*, i.e. $f_i(\mathbf{x}) = C$
D *discrete*, same as restricted values of C
S *simple (axis-aligned) linear functions*, i.e. $f_i(\mathbf{x}) = b_i + w_i \times x_j$, where $j \in \{1, 2 \dots n\}$ and $b_i, w_i \in \mathbb{R}$
L *linear*, i.e. $f_i(\mathbf{x}) = b_i + \mathbf{w}_i^T \mathbf{x}$
B *bilinear*, i.e. $f_i(\mathbf{x}) = x_1 \times x_2 + x_2 \times x_3$
Q *quadratic*, i.e. $f_i(\mathbf{x}) = b_i + \mathbf{w}_i^T \mathbf{x} + \mathbf{x}^T M_i \mathbf{x}$, where $M_i \in \mathbb{R}^{n \times n}$
P *polynomial* of order m, i.e. $f_i(\mathbf{x}) = \sum_{j_1=1}^{m} \cdots \sum_{j_n=1}^{m} w_{i,j_1,\dots j_n} \prod_{k=1}^{n} x_k^{j_k}$
N *general nonlinear*, i.e. $f(\mathbf{x}) = e^{b_i + \mathbf{w}_i^T \mathbf{x}}$.

Analogously, we consider analogous functions over integer (I) domains $\mathcal{X}$ or $\mathcal{Y}$, or mixed discrete-continuous (M) domains. $\mathcal{P}_i$ can be characterized similarly based on whether the constraint set defining it is constant, linear, bilinear, etc.

We also introduce a convenient notation to describe general PW functions by concatenating the expressivity classes of their constraints $\mathcal{P}_i$ and case functions f_i. For example, PWS-C describes a piecewise function on $\mathbb{R}^n$ with simple (axis-aligned) linear constraints and constant values, while PWS-L describes a function with similar constraints but linear values. We can also append the number of cases K to the notation, i.e. PWS1-L describes a piecewise linear function with $K = 1$ case conditions. More generally, for PWL, PWP or PWN, constraints could also be logical conjunctions of other simpler constraints.

2.2 Mathematical Programming

Given a function $f : \mathcal{X} \to \mathcal{Y}$ and Boolean case condition $\mathcal{P} : \mathcal{X} \to \{0, 1\}$, the *mathematical program* is defined as:

$$\min_{x \in \mathcal{X}} \quad f(x) \qquad \text{s.t.} \quad \mathcal{P}(x) = 1.$$

Important expressivity classes of *mixed-integer program* (MIP) optimization problems (i.e. those with discrete and continuous decision variables) include:

MILP *mixed-integer linear*: f and $\mathcal{P}$ both in L
MIBCP *mixed-integer bilinearly constrained*: f is in L and $\mathcal{P}$ is in B
MIQP *mixed-integer quadratic*: f is in Q and $\mathcal{P}$ is in L
QCQP *quadratically constrained quadratic*: f and $\mathcal{P}$ both in Q
PP *polynomial*: f and $\mathcal{P}$ both in P
MINLP *mixed-integer nonlinear*: f and $\mathcal{P}$ both in N.

Branch-and-bound [24] solvers are commonly used to maintain upper and lower bounds on the minimal objective value of any linear or nonlinear MIP, whose difference is the so-called *optimality gap*; when this gap is zero, an optimal solution has been found. Moreover, some packages such as Gurobi, which we use to perform the necessary compilations in CGPO in our experiments, also support a variety of nonlinear mathematical operations via piecewise-linear approximation [11].

2.3 Discrete-Continuous Markov Decision Processes

A *Discrete-Continuous Markov decision process* (DC-MDP) is a tuple of the form $\langle \mathcal{S}, \mathcal{A}, P, r \rangle$, where $\mathcal{S}$ is a set of states, and $\mathcal{A}$ is a set of actions or controls. $\mathcal{S}$ and $\mathcal{A}$ may be discrete, continuous, or mixed. $P(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ is the probability of transitioning to state $\mathbf{s}'$ immediately upon choosing action $\mathbf{a}$ in state $\mathbf{s}$, and $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ is the corresponding reward received. We assume that r is uniformly bounded, i.e. there exists $B < \infty$ such that $|r(\mathbf{s}, \mathbf{a}, \mathbf{s}')| \leq B$ holds for all $\mathbf{s}, \mathbf{a}, \mathbf{s}'$.

Given a fixed planning horizon of length T, the *value* of a (open-loop) *plan* $\alpha = [\mathbf{a}_1, \dots \mathbf{a}_T] \in \mathcal{A}^T$ starting in state $\mathbf{s}_1$ is

$$V(\alpha, \mathbf{s}_1) = \mathbb{E}_{\mathbf{s}_{t+1} \sim P(\mathbf{s}_t, \mathbf{a}_t, \cdot)} \left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \right].$$

A (closed-loop) *policy* $\pi = [\pi_1, \dots \pi_T]$ is a sequence of mappings $\pi_t : \mathcal{S} \to \mathcal{A}$, whose value is defined as[4]

$$V(\pi, \mathbf{s}_1) = \mathbb{E}_{\mathbf{s}_{t+1} \sim P(\mathbf{s}_t, \pi_t(\mathbf{s}_t), \cdot)} \left[\sum_{t=1}^{T} r(\mathbf{s}_t, \pi_t(\mathbf{s}_t), \mathbf{s}_{t+1}) \right].$$

Dynamic programming approaches such as value and policy iteration can compute an optimal horizon-dependent policy π^*_H [29], but do not directly apply to DC-MDPs with infinite or continuous state or action spaces.

Our goal is to compute an optimal stationary[5] policy π^* that minimizes the error in value relative to π^*_H over all initial states of interest $\mathcal{S}_1 \subseteq \mathcal{S}$, and all stationary policies $\mathit{\Pi}$

$$\pi^* \in \arg\min_{\pi \in \mathit{\Pi}} \max_{\mathbf{s}_1 \in \mathcal{S}_1} \left[V(\pi^*_H, \mathbf{s}_1) - V(\pi, \mathbf{s}_1) \right]. \tag{1}$$

[4] Deterministic (time-dependent) policies are sufficient for finite-horizon control [29].
[5] We focus on stationary policies for notational convenience; the extension is trivial.

In practical applications, the policy class is often restricted to function approximations $\tilde{\Pi} \subset \Pi$. A variety of planning approaches can compute plans $\tilde{\pi}^* \in \tilde{\Pi}$ for this problem, including straight-line planning that scales well in practice but does not learn policies [30,42]. Reactive policy optimization [10,23] and variants of MCTS [37] that learn neural network policies cannot provide concrete error bounds on policy performance nor do they facilitate worst-case analysis, or ease of interpretation of learned compact policy classes as we provide.

3 Methodology

We begin with a derivation of CGPO for general DC-MDPs in the deterministic setting. We then derive CGPO in the more general stochastic setting using chance constrained optimization. We end with a discussion of problem class complexity and convergence. A worked example illustrating the overall execution of CGPO is provided in the appendix.

3.1 Constraint Generation for Deterministic DC-MDPs

We assume π can be compactly identified by a vector $\mathbf{w} \in \mathcal{W}$ of decision variables, and we use the shorthand $V(\mathbf{w}, \mathbf{s}_1)$ to denote the value $V(\pi_{\mathbf{w}}, \mathbf{s}_1)$ of policy $\pi_{\mathbf{w}}$ parameterized by $\mathbf{w}$. We focus our attention on approximate policy sets $\tilde{\Pi}_{\mathcal{W}} = \{\pi_{\mathbf{w}} : \mathbf{w} \in \mathcal{W}\}$ enumerated by a compact set of parameters $\mathcal{W}$.

First, observe that for every possible initial state $\mathbf{s}_1$, there exists a fixed optimal plan $\alpha^* = [\mathbf{a}_1^*, \dots \mathbf{a}_T^*]$ such that $V(\alpha^*, \mathbf{s}_1) = V(\pi_H^*, \mathbf{s}_1)$. On the other hand, since we only have access to an expressivity class of approximate policies $\tilde{\Pi}_{\mathcal{W}}$, the error $\varepsilon(\mathbf{w}, \mathbf{s}_1)$ of $\pi_{\mathbf{w}}$ in $\mathbf{s}_1$ relative to $V(\pi_H^*, \mathbf{s}_1)$ according to (1) is

$$\varepsilon(\mathbf{w}, \mathbf{s}_1) \geq \max_{\alpha \in \mathcal{A}^T} V(\alpha, \mathbf{s}_1) - V(\mathbf{w}, \mathbf{s}_1),$$

and thus the worst-case error $\varepsilon(\mathbf{w})$ of $\mathbf{w}$ is

$$\varepsilon(\mathbf{w}) \geq \max_{\mathbf{s}_1 \in \mathcal{S}_1} \max_{\alpha \in \mathcal{A}^T} \left[V(\alpha, \mathbf{s}_1) - V(\mathbf{w}, \mathbf{s}_1) \right]. \tag{2}$$

However, since we seek the approximate optimal policy $\tilde{\pi}^* \in \tilde{\Pi}_{\mathcal{W}}$, we can directly minimize (2) over $\mathcal{W}$, obtaining the *infinitely-constrained mixed-integer program*:

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W},\, \varepsilon \in [0,\infty)} \quad & \varepsilon \\ \text{s.t.} \quad & \varepsilon \geq \max_{\mathbf{s}_1 \in \mathcal{S}_1} \max_{\alpha \in \mathcal{A}^T} \left[V(\alpha, \mathbf{s}_1) - V(\mathbf{w}, \mathbf{s}_1) \right]. \end{aligned} \tag{3}$$

However, the constraint is highly nonlinear, turning (3) into a bilevel program and making analysis of this particular formulation difficult. Instead, since the max's must hold for all states and actions, we can rewrite the problem (3) as:

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W},\, \varepsilon \in [0,\infty)} \quad & \varepsilon \\ \text{s.t.} \quad & \varepsilon \geq V(\alpha, \mathbf{s}_1) - V(\pi, \mathbf{s}_1), \quad \forall \alpha \in \mathcal{A}^T,\, \mathbf{s}_1 \in \mathcal{S}_1. \end{aligned} \tag{4}$$

The goal is therefore to solve (4), which is still infinitely-constrained when $\mathcal{S}$ or $\mathcal{A}$ are infinite spaces. Instead, we solve it by splitting the min over $\mathcal{W}$ and the max over $(\alpha, \mathbf{s}_1)$ into two problems and apply *constraint generation* [7,12].

Specifically, starting with a fixed arbitrary scenario $(\mathbf{s}_1, \alpha)$, we form the constraint set $\mathcal{C} = \{(\mathbf{s}_1, \alpha)\}$, and then solve the following two problems:

Outer Solve (4) within the set of finite constraints $\mathcal{C}$ to obtain $\mathbf{w}^* \in \mathcal{W}$.
Inner Solve (2) $\text{argmax}_{\mathbf{s}_1 \in \mathcal{S}_1, \alpha \in \mathcal{A}^T} [V(\alpha, \mathbf{s}_1) - V(\mathbf{w}^*, \mathbf{s}_1)]$, for the highest-error scenario for $\mathbf{w}^*$, and append it to $\mathcal{C}$.

These two steps are repeated until the constraint added to the outer problem is no longer binding, i.e. $V(\alpha^*, \mathbf{s}_1^*) - V(\mathbf{w}^*, \mathbf{s}_1^*) \leq 0$, since the solution of the outer problem will not change with the addition of the new constraint. We name our approach *Constraint-Generation Policy Optimization* (CGPO). *In retrospect, we can view CGPO as a policy iteration algorithm where the inner problem adversarially critiques the policy with a worst-case trajectory and the outer problem improves the policy w.r.t. all critiques.*

Remarks Upon termination, CGPO guarantees an optimal (lowest error) policy within the specified policy class (Theorem 1). While we cannot provide a general finite-time guarantee of termination with an optimal policy, CGPO is an anytime algorithm that provides a best policy and a bound on performance at each iteration. At each iteration, the solution to the inner problem allows analysis of the worst-case trajectory with respect to π (Fig. 7), and generates a counterfactual explanation of what actions should have been made.

3.2 Chance Constraints for Stochastic DC-MDPs

Chance-constrained optimization [4,16,25] allows us to derive high probability intervals on stochastic MDP transitions and reduce our solution to a robust optimization problem with probabilistic performance guarantees. To achieve this, we will require that the state transition function $P(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ has a natural reparameterization as a deterministic function g of $(\mathbf{s}, \mathbf{a})$ and some exogenous independent and identically distributed noise variable ξ with density $q(\cdot)$ on support Ξ, e.g. $\mathbf{s}' = g(\mathbf{s}, \mathbf{a}, \xi)$ [10,26]. Given a threshold $p \in (0, 1)$ close to 1, we also assume the existence of a computable interval $\Xi_p = [\xi_l, \xi_u]$ such that $\mathbb{P}(\xi_l \leq \xi \leq \xi_u) \geq p$.

Thus, we can repeat the derivation of (3) by considering the worst case not only over $\mathbf{s}_1$, but also over possible noise variables $\xi_{1:T} = [\xi_1, \ldots \xi_T] \in \Xi_p^T$. The final problem is:

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W},\, \varepsilon \in [0,\infty)} \quad & \varepsilon \\ \text{s.t.} \quad & \varepsilon \geq \max_{\mathbf{s}_1 \in \mathcal{S}_1} \max_{\xi_{1:T} \in \Xi_p^T} \max_{\alpha \in \mathcal{A}^T} \left[V(\alpha, \mathbf{s}_1, \xi_{1:T}) - V(\mathbf{w}, \mathbf{s}_1, \xi_{1:T}) \right], \end{aligned} \tag{5}$$

where $V(\cdot, \mathbf{s}_1, \xi_{1:T})$ corresponds to the total reward of the policy or plan accumulated over trajectory $\xi_{1:T}$ starting in $\mathbf{s}_1$. In the stochastic setting, ε only

holds with probability p^T for a horizon T problem. Experimentally, we found that choosing p such that $p^T = P$, where P is a desired probability bound on the full planning trajectory, was sufficient[6].

Once again, (5) can be reformulated as:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathcal{W},\,\varepsilon\in[0,\infty)} \quad \varepsilon \\ &\text{s.t.} \quad \varepsilon \geq V(\alpha, \mathbf{s}_1, \xi_{1:T}) - V(\mathbf{w}, \mathbf{s}_1, \xi_{1:T}), \quad \forall \alpha \in \mathcal{A}^T,\, \mathbf{s}_1 \in \mathcal{S}_1,\, \xi_{1:T} \in \Xi_p^T. \end{aligned} \tag{6}$$

Therefore, we can again apply constraint generation to solve this problem in two stages, in which the inner optimization produces not only a worst-case initial state and action sequence, but also the disturbances $\xi^*_{1:T}$ that reproduce all future worst-case state realizations $\mathbf{s}_2, \ldots \mathbf{s}_T$. The overall workflow of the computations is summarized as Algorithm 1.

Algorithm 1 Constraint-Generation Policy Optimization (CGPO)

Initialize $p \in (0,1)$, Ξ_p, $\mathbf{s}_1 \in \mathcal{S}_1$, $\xi_{1:T} \in \Xi_p^T$, $\alpha \in \mathcal{A}^T$
Set $t = 0$ and $\mathcal{C}_0 = \{(\mathbf{s}_1, \xi_{1:T}, \alpha)\}$
Step 1: Solve the **outer problem**:

$$(\mathbf{w}_t^*, \varepsilon_t^*) \in \operatorname{argmin}_{\mathbf{w}\in\mathcal{W},\,\varepsilon\in[0,\infty)} \varepsilon$$
$$\text{s.t.} \quad \varepsilon \geq V(\alpha, \mathbf{s}_1, \xi_{1:T}) - V(\mathbf{w}, \mathbf{s}_1), \quad \forall\, (\mathbf{s}_1, \xi_{1:T}, \alpha) \in \mathcal{C}_t$$

Step 2: Solve the **inner problem**:

$$(\mathbf{s}_1^*, \xi^*_{1:T}, \alpha^*) \in \operatorname{argmax}_{\mathbf{s}_1\in\mathcal{S}_1,\,\xi_{1:T}\in\Xi_p^T,\,\alpha\in\mathcal{A}^T} \left[V(\alpha, \mathbf{s}_1, \xi_{1:T}) - V(\mathbf{w}_t^*, \mathbf{s}_1, \xi_{1:T})\right]$$

Step 3: Check convergence:
if $V(\mathbf{w}_t^*, \mathbf{s}_1^*, \xi^*_{1:T}) \geq V(\alpha^*, \mathbf{s}_1^*, \xi^*_{1:T})$ (or $\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\| < \delta$) **then**
  Terminate with policy $\pi_{\mathbf{w}_t^*}$ and error ε_t^*
else
  Set $t = t + 1$, $\mathcal{C}_{t+1} = \mathcal{C}_t \cup \{(\mathbf{s}_1^*, \xi^*_{1:T}, \alpha^*)\}$ and **go to Step 1**
end if

Remark Algorithm 1 may not terminate in general, unless convexity assumptions are placed on $V(\alpha, \mathbf{s})$ or $V(\mathbf{w}, \mathbf{s})$. Thus, in typical applications, one would terminate when $\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\| < \delta$ for chosen hyper-parameter δ. However, our experiments have shown empirically that Algorithm 1 terminates with an exact optimal solution for most policy classes considered, at which point both the return and error curves "plateau" (cf. Fig. 3).

3.3 Optimality of CGPO

Under mild conditions, if CGPO terminates at some iteration t, then $\mathbf{w}_t^*$ is optimal in $\mathcal{W}$ (with probability at least p^T in the stochastic case).

[6] Hence, we arbitrarily chose $p = 0.995$ for our empirical evaluations.

Theorem 1. *If $\mathcal{S}_1$, $\mathcal{A}$, Ξ_p and $\mathcal{W}$ are non-empty compact subsets of Euclidean space, $V(\alpha, \mathbf{s}, \xi_{1:T})$ and $V(\mathbf{w}, \mathbf{s}, \xi_{1:T})$ are continuous, and CGPO terminates at iteration t, then $\mathbf{w}_t^*$ is optimal for problem (6).*

Proof. In the notation of [7], we define $X^0 = [0, 2BT]$, $X = \mathcal{W}$, $Y = \mathcal{A}^T \times \mathcal{S}_1 \times \Xi_p^T$. Making the change of variable $\varepsilon = x_0 \in X^0$, $x = \mathbf{w}$, and $y = (\alpha, \mathbf{s}_1, \xi_{1:T}) \in Y$, and defining the continuous function $f(x, y) = \varepsilon(\alpha, \mathbf{s}_1, \xi_{1:T}) = V(\alpha, \mathbf{s}_1, \xi_{1:T}) - V(\mathbf{w}, \mathbf{s}_1, \xi_{1:T})$, the problem (6) is equivalent to the problem:

$$\min_{(x_0, x) \in X^0 \times X} x_0 \qquad \text{s.t.} \quad f(x, y) - x_0 \leq 0, \quad \forall y \in Y$$

as discussed on p. 262 of [7]. Similarly, Algorithm 1 can be reparameterized as Algorithm 2.1 in the aforementioned paper, with $x_t = \mathbf{w}_t^*$. Thus, the assumption of Theorem 2.1 in [7] holds and $\mathbf{w}_t^*$ is a solution of (6) as claimed. □

Remark Termination of CGPO guarantees that $\mathbf{w}_t^*$ is optimal in $\tilde{\Pi}_{\mathcal{W}}$, and ε_t^* is the corresponding optimality gap. Moreover, if $\varepsilon_t^* = 0$ and the MDP is deterministic, then $\pi_{\mathbf{w}_t^*}$ is the optimal policy in Π. This implies that the gap estimate can be used as a principled way to assess the suitability of different policy classes.

3.4 Problem Expressivity Class Analysis

	Dynamics / Reward		
Policy	L	P	N
PWS-{C, D}	MILP	PP	MINLP
PWL-{C, D}	MILP/MIBCP	PP	MINLP
S, L	MILP/PP	PP	MINLP
PW{S,L}-{S,L}	MILP/PP	PP	MINLP
PW{S,L,P}-P	PP	PP	MINLP
Q	PP	PP	MINLP
PWN-N	MINLP	MINLP	MINLP

Table 1. Problem classes of the inner/outer optimization problems in CGPO for different expressivity classes of dynamics/reward (columns) and policies (rows).

In Table 1, we present the relationship between the expressivity classes of policies and state transition dynamics and the corresponding classes of the inner and outer optimization problems. The policy classes of interest include PWS-C, PWL-C, PWS-L, PWL-L, PWN-N and the different variants of piecewise polynomial policies. Meanwhile, expressivity classes for state dynamics and reward include linear, polynomial and general nonlinear functions (their piecewise counterparts generally fall under the same expressivity classes, and are excluded for brevity). Interestingly, as shown in the appendix, when the policy and state dynamics are both linear, the outer problem is a PP. In a similar vein, a PWL-C

policy and linear dynamics result in a MIBCP outer problem due to the bilinear interaction between successor state decision variables and policy weights in the linear conditions.

Our experiments in the next section empirically evaluate PWS-C and PWS-L policies with linear dynamics and Q policies with nonlinear dynamics. This requires solving mixed-integer problems with large numbers of decision variables ranging from MILP to PP to MINLP.

4 Empirical Evaluation

We empirically validate the performance of CGPO on several MDPs, aiming to answer the following questions:

Q1 Does CGPO recover exact solutions when the ideal policy class is known? How does it perform if the optimal policy class is not known?
Q2 How do different policy expressivity classes perform for each problem?
Q3 Does the worst-case analysis provide further insight about a policy?

4.1 Domain Descriptions

To answer these questions, we evaluate on linear Inventory, linear Reservoir, and nonlinear VTOL (vertical take-off and landing) domains as summarized below. Inventory has provably optimal PWS-S policies, whereas no optimal policy class is known explicitly for Reservoir. A public GitHub repository[7] allows reproduction of all results and application of CGPO to arbitrary domains, and also contains an appendix with experimental details and additional results.

Linear Inventory State s_t describes the discrete level of stock available for a single good, action $a_t \in [0, B]$ is the discrete reorder quantity, and demand ξ_t is stochastic and distributed as discrete uniform:

$$s_{t+1} = s_t + a_t - \lfloor \xi_t \rfloor, \quad \xi_t \sim \text{Uniform}(L, U).$$

We permit backlogging of inventories, represented as negative s_t. We define costs C, P and S which represent, respectively, the purchase cost, excess inventory cost and shortage cost, and the reward function can be written as

$$r(\mathbf{s}_{t+1}) = -C \times a_t - P \times (s_{t+1})_+ - S \times (-s_{t+1})_+,$$

where $(\cdot)_+ = \max[0, \cdot]$. We set $B = 10$, $C = 0.5$, $P = S = 2$, $L = 2$, $U = 6$. If $P > C$ and $T = \infty$, then a PWS-S policy is provably optimal (otherwise if $T < \infty$, then it may be non-stationary). A planning horizon of $T = 8$ is used. For this domain and reservoir below, we focus on learning factorized piecewise policies, i.e. C, S, PWS-C and PWS-S.

[7] https://github.com/pyrddlgym-project/pyRDDLGym/tree/GurobiCompilerBranch/

Linear Reservoir The goal is to manage the water level in a system of interconnected reservoirs. State $s_{t,r}$ represents the continuous water level of reservoir r with capacity M_r, action $a_{t,r} \in [0, s_{t,r}]$ is the continuous amount of water released, and rainfall $\xi_{t,r}$ is a truncated normally-distributed random variable. Each reservoir r is connected to a set $U(r)$ of upstream reservoirs, thus:

$$s_{t+1,r} = \min\Big[M_r, \Big(s_{t,r} + (\xi_{t,r})_+ - a_{t,r} + \sum_{d \in U(r)} a_{t,d}\Big)_+\Big], \quad \xi_{t,r} \sim \mathcal{N}(m_r, v_r).$$

Reward linearly penalizes any excess water level above H_r and below L_r

$$r(\mathbf{s}_{t+1}) = \sum_{r=1}^{N} l_r(L_r - s_{t+1,r})_+ + h_r(s_{t+1,r} - H_r)_+.$$

Our experiment uses a two-reservoir system with the following values for reservoir 1: $L_1 = 20, H_1 = 80, M_1 = 100, m_1 = 5, v_1 = 5, U(1) = \emptyset$, and the following values for reservoir 2: $L_2 = 30, H_2 = 180, M_2 = 200, m_2 = 10, v_2 = 10, U(2) = \{1\}$. Costs are identical for all reservoirs and are set to $l_r = -10, h_r = -100$. We use $T = 10$.

Nonlinear VTOL The goal is to balance two masses on opposing ends of a long pole. The state consists of the angle θ_t and angular velocity ω_t of the pole, and the action is the force $F_t \in [0, 1]$ applied to the mass:

$$\begin{aligned}
\theta_{t+1} &= \max\left[-\sin(h/l_1), \min\left[\sin(h/l_2), \theta_t + \tau\omega_t\right]\right] \\
\omega_{t+1} &= \omega_t + \frac{\tau}{J}\left(9.8(m_2 l_2 - m_1 l_1)\cos\theta_t + 150 l_1 F_t\right) \\
J &= m_1 l_1^2 + m_2 l_2^2 \\
\text{s.t.} \quad & l_1 m_1 \geq l_2 m_2, \quad l_1 > l_2 > h, \quad 0 \leq F_t \leq 1.
\end{aligned}$$

Time is discretized into intervals of $\tau = 0.1$ seconds, and we set $T = 6$. The reward penalizes the difference between the pole angle at epoch t and a target angle θ_{target}

$$r(F_t, \theta_{t+1}, \omega_{t+1}) = -|\theta_{t+1} - \theta_{target}|.$$

We use $l_1 = 1, l_2 = 0.5, m_1 = 10, m_2 = 1, h = 0.4, g = 9.8$. We optimize for nonlinear quadratic policies.

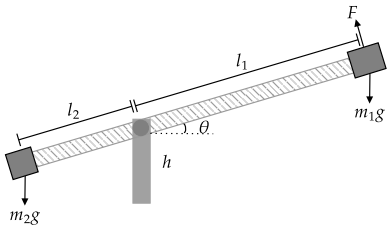


Fig. 2. Nonlinear VTOL system.

Nonlinear Intercept To demonstrate an example of discrete (Boolean) action policy optimization over nonlinear continuous state dynamics with independently moving but interacting projectiles, we turn to the Intercept problem [33]. Space restrictions require us to relegate details and results to the appendix, but we remark here that CGPO is able to derive an optimal policy.

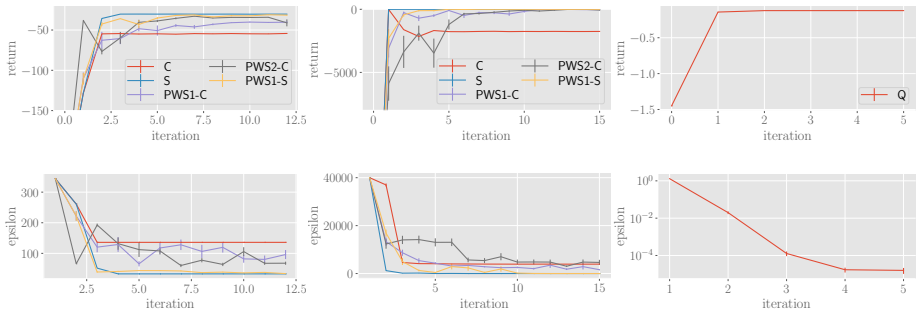
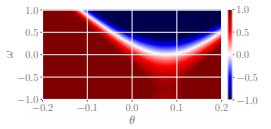


Fig. 3. Reservoir (left), Inventory (middle), VTOL (right): Simulated return (top row) and worst-case error ε^* (bottom row) over 100 roll-outs as a function of the number of iterations of constraint generation. Bars represent 95% confidence intervals calculated using 10 independent runs of CGPO.

$$release_1 = \begin{cases} 33.23 & \text{if } 65.06 \leq level_1 \leq 100 \\ 1.08 & \text{otherwise} \end{cases}$$

$$release_2 = \begin{cases} 72.91 & \text{if } 100 \leq level_2 \leq 200 \\ 0.2 & \text{otherwise} \end{cases}$$

$$reorder = \begin{cases} 5 - stock & \text{if } -1 \leq stock \leq 2 \\ 3 - stock & \text{otherwise} \end{cases}$$



$$F = 0.6 - 15.4\theta - 2.3\omega + 100\theta^2 - 1.5\omega^2$$

Fig. 4. Reservoir (left), Inventory (middle), VTOL (right): Examples of optimal policies computed at the end of CGPO.

Initial State Bounds $\mathcal{S}_1$ For Reservoir, the initial state bounds are set to $[50, 100]$ for reservoir 1 and $[100, 200]$ for reservoir 2. For Inventory, the bounds are $[0, 2]$, and for VTOL and Intercept they are fixed to the initial state of the system.

4.2 Empirical Results

Fig. 3 compares the empirical (simulated) returns and error bounds ε^* (optimal objective value of the inner problem) across different policy classes. Fig. 4 illustrates examples of policies learned in a typical run of CGPO. As hypothesized, CGPO can recover exact (s, S) control policies for Inventory in the PWS-S class, which together with S policies achieve the lowest error and best return across all policy classes. Interestingly, as Fig. 3 shows, C and PWS-C policies perform relatively poorly even for $K = 2$, but as expected, the error decreases as the expressivity of the policy class, and in particular the number of cases, increases. This is intuitive, as Fig. 4 shows, the reorder quantity is strongly linearly dependent on the current stock, which is not easily represented as PWS-C. In constrast, on Reservoir the class of PWS-C policies perform comparatively well for $K \geq 1$, and achieve (near)-optimality despite requiring a larger number of iterations. As expected, policies typically release more water as the water level

approaches the upper target bound. Finally, the optimal policy for VTOL applies a large upward force when the angle or angular velocity are negative, with relative importance being placed on angle. As Fig. 4 (right) shows, the force is generally greater when either the angle is below the target angle or the angular velocity is negative, and equilibrium is achieved by applying a modest upward force between the initial angle (0.1) and the target angle (0).

Fig. 5 and 6 provide examples of policies learned in other expressivity classes for Inventory and Reservoir problems, respectively. In all cases, policies remain intuitive and easy to explain even as the complexity increases. For instance, PWS2-C policies for Inventory order more inventory as the stock level decreases, while for Reservoir, the amount of water released is increasing in the water level.

$$order = 3 \qquad order = 4 - stock \qquad order = \begin{cases} 8 & \text{if } -13 \leq stock \leq -2 \\ 2 & \text{otherwise} \end{cases} \qquad order = \begin{cases} 6 & \text{if } -10 \leq stock \leq -1 \\ 1 & \text{if } 2 \leq stock \leq 2 \\ 3 & \text{otherwise} \end{cases}$$

Fig. 5. Inventory: Examples of C, S, PWS1-C and PWS2-C (left to right) policies computed by CGPO.

$$\begin{aligned} release_1 &= 19.73 \\ release_2 &= 55.87 \end{aligned} \qquad \begin{aligned} release_1 &= level_1 - 20 \\ release_2 &= 0.93 \times level_2 - 29.74 \end{aligned}$$

$$release_1 = \begin{cases} 1.39 & \text{if } 30.62 \leq level_1 \leq 68.66 \\ 40.71 & \text{if } 70.93 \leq level_1 \leq 100 \\ 10.622 & \text{otherwise} \end{cases}$$

$$release_2 = \begin{cases} 80.78 & \text{if } 159.93 \leq level_2 \leq 200 \\ 38.69 & \text{if } 144.21 \leq level_2 \leq 159.11 \\ 18.25 & \text{otherwise} \end{cases}$$

$$release_1 = \begin{cases} level_1 - 50 & \text{if } 20 \leq level_1 \leq 100 \\ 2 \times level_1 - 100 & \text{otherwise} \end{cases}$$

$$release_2 = \begin{cases} level - 2 - 37.8 & \text{if } 38.9 \leq level_2 \leq 200 \\ 1.17 \times level_2 - 235.5 & \text{otherwise} \end{cases}$$

Fig. 6. Reservoir: Examples of C, S, PWS2-C and PS1-S (left to right, top to bottom) policies computed by CGPO.

4.3 Ablations

Worst-Case Analysis Fig. 7 plots the state trajectory, actions and rainfall that lead to worst-case performance of C and S policies at the last iteration of CGPO on Reservoir. Here, we observe that the worst-case performance for the optimal C policy occurs when the rainfall is low and both reservoirs become empty. Given that the cost of overflow exceeds underflow, this is expected as the C policy must release enough water to prevent high-cost overflow events during high rainfall at the risk of water shortages during droughts. In contrast, the optimal S policy maintains safe water levels even in the worst-case scenario, since it avoids underflow and overflow events by releasing water (linearly) proportional to the water level. Similar to Reservoir, a non-trivial worst-case scenario is identified

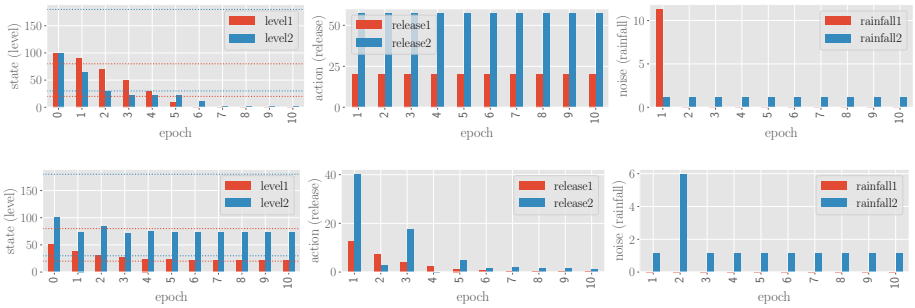


Fig. 7. Reservoir: Worst-case state trajectory (left), actions (middle), and noise (right) for C (top) and S policies (bottom) computed by CGPO. The more expressive S policy provides much more stable water levels.

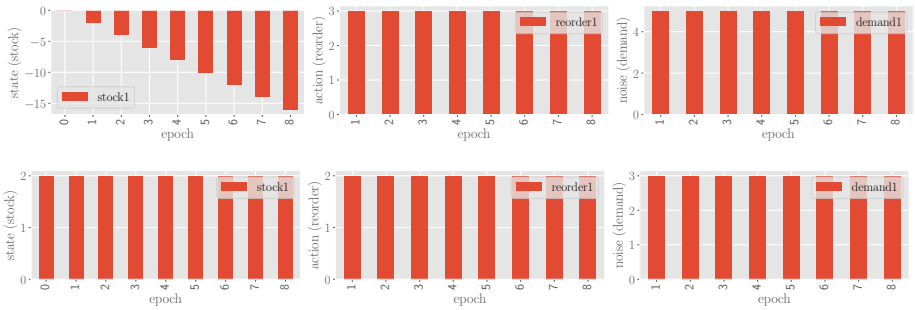


Fig. 8. Inventory: Worst-case state trajectory (left), actions (middle), and disturbances/noise (right) for C (top) and S policies (bottom) computed by CGPO.

by CGPO for Inventory (Fig. 8) for optimal C policies but not S policies. This scenario corresponds to very high demand that exceeds the constant reorder quantity, causing stock-outs. For VTOL (Fig. 9), there is no worst-case scenario with non-zero cost upon convergence at the final iteration of CGPO (bottom row). We have also shown the worst-case scenario computed at the first iteration prior to achieving policy optimality, where a worst-case scenario corresponds to no force being applied causing the system to eventually lose balance.

Analysis of Problem Size To understand how the problem sizes in CGPO scale across differing expressivity classes of policies and dynamics, Table 2 shows the number of decision variables and constraints in the MIPs at the last iteration of constraint generation. For VTOL with a quadratic (Q) policy, the number of variables in the inner and outer problems are 0 / 0 / 212 and 0 / 0 / 512, respectively, while the number of constraints are 114 / 18 / 72 and 270 / 95 / 155. The number of decision variables/constraints are fixed in the inner problem and grow linearly with iterations in the outer problem. The number of variables

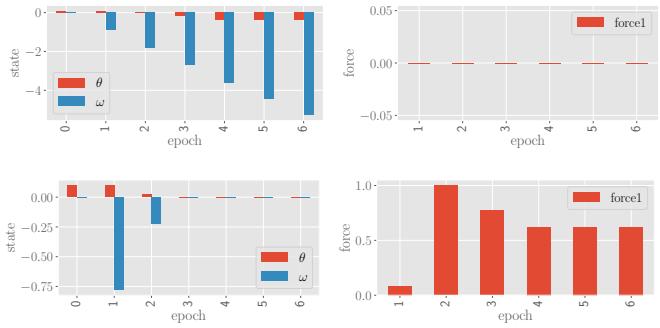


Fig. 9. VTOL: Worst-case state trajectory (top), and actions (bottom) for Q policies. First (last) column corresponds to the first (last) iterations of CGPO.

	Inventory		Reservoir	
Policy	Inner	Outer	Inner	Outer
C	8 / 121 / 72	96 / 673 / 385	20 / 0 / 822	300 / 0 / 5823
PWS-S	32 / 129 / 72	384 / 772 / 387	80 / 0 / 842	1200 / 0 / 7133
C	160 / 0 / 64	780 / 0 / 480	540 / 0 / 320	4035 / 0 / 2400
PWS-S	168 / 0 / 112	709 / 168 / 1056	560 / 0 / 440	3797 / 540 / 4200

Table 2. Number of binary/integer/real-valued decision variables in the inner and outer optimization problems at the last iteration of CGPO (top half), and the number of linear/quadratic/general constraints (bottom half).

and constraints do not grow significantly with the policy class expressiveness. Please note that we do not report runtimes due to the highly stochastic and unpredictable nature of the runs.

5 Conclusion

We presented CGPO, a novel bilevel mixed-integer programming formulation for DC-MDP policy optimization in predefined expressivity classes with bounded optimality guarantees. **To the best of our knowledge, CGPO is the first algorithm capable of providing bounded error guarantees for a broad range of DC-MDPs and policy classes.** We used constraint generation to decompose CGPO into an inner problem that produces an adversarial worst-case constraint violation (i.e., trajectory), and an outer problem that improves the policy w.r.t. these trajectories. Across diverse domains and policy classes, we showed that the learned policies (some provably optimal) and their worst-case performance/failure modes were easy to interpret, while maintaining a manageable number of variables and constraints. To overcome the significant computational demands required by CGPO to tackle larger problems, future work should leverage tools from reinforcement learning to approximately solve the inner/outer problems.

6 Appendix

In this supplement to the main paper, we provide further experimental details for successful implementation of CGPO. We also provide a worked example illustrating the mechanics of CGPO, additional convergence/performance results and worst-case analysis for the domains that were left out of the main text due to space limitations. We conclude with a discussion of related work.

6.1 Worked Example

Domain Description Let the state s_t denote the continuous position of a particle on $\mathbb{R}$ at epoch $t = 1, 2$, and let the action be a continuous unbounded displacement a_t applied to the particle, so that the state updates according to $s_{t+1} = s_t + a_t$. Thus, $\mathcal{S} = \mathcal{A} = \mathbb{R}$ and suppose further that $\mathcal{S}_1 = [0, 5]$. If the target position is designated as 10, we formulate the reward as $r(s_{t+1}) = -|s_{t+1} - 10|$, and thus the value function is $V(a_1, s_1) = -|s_1 + a_1 - 10|$.

The Outer Problem For ease of illustration, we focus our attention on linear policies $\pi_{w,b}$ of the form $a_t = b + ws_t$, where (w, b) are real-valued decision variables. The goal is to find the optimal policy parameters

$$\min_{(w,b)\in\mathbb{R}^2} \max_{s_1\in[0,5]} \max_{a_1\in\mathbb{R}} [V(a_1, s_1) - V((w, b), s_1)],$$

where $V((w, b), s_1) = -|s_1 + b + ws_1 - 10|$ is the value of the linear policy. However, this problem is not directly solvable due to the inner max, but it can be written as:

$$\min_{(w,b)\in\mathbb{R}^2,\, \varepsilon\in[0,\infty)} \varepsilon \qquad \text{s.t.} \quad \varepsilon \geq V(a_1, s_1) - V((w, b), s_1),$$

where constraints are enumerated by all possible (s_1, a_1) pairs. Since the number of constraints is infinite for continuous states or actions, it suggests a constraint generation approach, where a large set of diverse (s_1, a_1) pairs are used to "build up" the constraint set one at a time, decoupling the overall problem into a sequence of solvable MIPs. However, a key question arises: **which state-action pairs (s_1, a_1) should be considered for inclusion into the outer problem?** While many different reasonable choices exist, a particularly logical choice is to select the state-action pairs on which the policy $\pi_{w,b}$ parameterized by (w, b) performs worst.

Solve the Outer Problem The outer problem thus maintains a finite subset of constraints from the infinitely-constrained problem. We begin with an arbitrary (s, a)-pair, which forms the first constraint of the outer problem. For illustration, we select $(s_1, a_1) = (0, 0)$, thus with the constraint

$$\varepsilon \geq V(0, 0) - V((w, b), 0) = -10 + |b - 10|,$$

the outer problem becomes:

$$\min_{w,b\in\mathbb{R},\,\varepsilon\in[0,\infty)} \varepsilon \qquad \text{s.t.} \quad \varepsilon \geq -10 + |b - 10|.$$

Recognizing that the minimum of the r.h.s. of the constraint is attained at $b^* = 10$, and w^* arbitrary, we select $(w^*, b^*) = (0, 10)$ as the solution of the outer problem with optimal $\varepsilon = 0$. Next, we will try to improve upon the policy by adding an additional constraint to the outer problem.

Solve the Inner Sub-Problem To accomplish this, we need to add a new scenario (s_1, a_1) in relation to which the linear policy with parameters $(w, b) = (0, 10)$ performs worst. Thus, we solve:

$$\max_{s_1\in[0,5],a_1\in\mathbb{R}} [V(a_1, s_1) - V((w, b), s_1)] = \max_{s_1\in[0,5],a_1\in\mathbb{R}} [-|s_1 + a_1 - 10| + s_1],$$

from which we find $(s_1^*, a_1^*) = (5, 5)$. The optimal value of this problem is positive, so it represents a potentially binding constraint in the outer problem. Thus, we could improve upon the policy by adding the corresponding constraint.

Solve the Outer Problem Again Adding the constraint to the outer problem:

$$\min_{(w,b)\in\mathbb{R}^2,\,\varepsilon\in[0,\infty)} \varepsilon \qquad \text{s.t.} \quad \varepsilon \geq -10 + |b - 10|, \quad \varepsilon \geq |b + 5w - 5|,$$

whose new solution can be found as $b^* = 10, w^* = -1$ with optimal $\varepsilon = 0$. Finally, the worst-case scenario for this new policy can be found from the inner sub-problem; one solution is $(s_1^*, a_1^*) = (0, 10)$ with value zero. This corresponds to a non-binding constraint, and thus we terminate with the optimal linear policy $\pi^*(s_1) = 10 - s_1$.

6.2 Linear Policy + Linear Dynamics Result in a Polynomial Class

Suppose both the policy and transition functions are linear in the state. Specifically, consider the transition function $s' = s + a$, and the policy $a = ws$. Then, defining s the initial state, s' the immediate successor state following s, and s'' the immediate successor state following s':

$$\begin{aligned}
s' &= s + a = s + ws = (w + 1)s \\
a' &= ws' = w(w + 1)s = O(w^2) \\
s'' &= s' + a' = (w + 1)s + w(w + 1)s = (w + 1)^2 s \\
a'' &= ws'' = w(w + 1)^2 s = O(w^3),
\end{aligned}$$

which are polynomial in w.

6.3 The Nonlinear Intercept Problem

Intercept is a nonlinear domain with Boolean actions. It involves two objects moving in continuous trajectories in a subset of $\mathbb{R}^2$, in which one object (e.g. missile, bird) flies in a parabolic arc across space towards the ground, and must be intercepted by a second object (e.g. anti-ballistic missile, predator) that is fired from a fixed position on the ground. While the problem is best described in continuous time, we study a discrete-time version of the problem. The state includes the position (x_t, y_t) of the missile at each decision epoch, as well as "work" variables indicating whether the interceptor has already been fired f_t as well as its vertical elevation i_t. Meanwhile, the action a_t is Boolean-valued and indicates whether the interceptor is fired at a given time instant.

The state transition model can be written as:

$$x_{t+1} = x_t + v_x, \qquad y_{t+1} = h - \frac{1}{2} g x_t^2,$$
$$f_{t+1} = f_t \vee a_t, \qquad i_{t+1} = \begin{cases} i_t & \text{if } f_t = 0 \\ i_t + v_y & \text{if } f_t = 1 \end{cases},$$

ignoring the gravitational interaction of the interceptor. Interception happens whenever the absolute differences between the coordinates of the missile and the interceptor are within δ, so the reward is

$$r(s_{t+1}) = \begin{cases} 1 & \text{if } f_{t+1} \wedge |x_{t+1} - i_x| \leq \delta \wedge |y_{t+1} - i_{t+1}| \leq \delta \\ 0 & \text{otherwise} \end{cases}.$$

We set $v_x = 0.1, h = 5, g = 9.8, i_x = 0.7$.

We study Boolean-valued policies with linear constraints (i.e., B, PWL-B) that take into account the position of the missile at each epoch, i.e. PWL1-B policies of the form

$$a_t(x_t, y_t) = \begin{cases} a_1 & \text{if } l \leq w_1 x_t + w_2 y_t \leq u \\ a_2 & \text{otherwise} \end{cases}$$

where $a_1, a_2 \in \{0, 1\}$ and l, u, w_1, w_2 are all tunable parameters.

6.4 Additional Implementation Details

Computing Environment We use the Python implementation of the Gurobi Optimizer (GurobiPy) version 10.0.1, build v10.0.1rc0 for Windows 64-bit systems, with an academic license. Default optimizer settings have been used, with the exception of NumericFocus set to 2 in order to enforce numerical stability, and the MIPGap parameter set to 0.05 to terminate when the optimality gap reaches 5%. To compute the tightest possible bounds on decision variables in the MIP compilation, we use interval arithmetic [21]. All experiments were conducted on a single machine with an Intel 10875 processor (base at 2.3 GHz, overclocked at 5.1 GHz) and 32 GB of RAM. A runtime cap of 2 hours was enforced for each run. However, we do not report runtimes due to the highly stochastic and unpredictable nature of the runs.

Domain Description Files Domains are described in *Relational Dynamic influence Diagram Language* (RDDL), a structured planning domain description language particularly well-suited for modelling problems with stochastic effects [31]. To facilitate experimentation, we wrote a general-purpose routine for compiling RDDL code into a Gurobi mixed integer program formulation using the `pyRDDLGym` interface [38].

Action Constraints One important issue concerns how to enforce constraints on valid actions. Two valid approaches include (1) explicitly constraining actions in RDDL constraints (state invariants and action preconditions) by compiling them as MIP constraints, or (2) implicitly clipping actions in the state dynamics (conditional probability functions in RDDL). Crucially, we found the latter approach performed better during optimization, since constraining actions inherently limits the policy class to a subset of weights that can only produce legal actions across the initial state space. This not only significantly restricts policies (i.e. for linear valued policies, the weights would be constrained to a tight subset where the output for every possible state would be a valid action) but also poses challenges for the optimization process, which is significantly complicated by these action constraints.

Encoding Constraints in Gurobi Mathematical operations, such as strict inequalities, cannot be handled explicitly in Gurobi. To perform accurate translation of such operations in our code-base, we used indicator/binary variables. For instance, to model the comparison $a > b$ for two numerical values a, b, we define a binary variable $y \in \{0, 1\}$ and error parameter $\epsilon > 0$ constrained as follows:

$$y == 1 \implies a \geq b + \epsilon, \qquad y == 0 \implies a \leq b.$$

In practice, ϵ is typically set larger than the smallest positive floating point number in Gurobi (around 10^{-5}), but is often problem-dependent. We derived optimal policies for all domains using $\epsilon = 10^{-5}$, except Intercept where we needed to use a larger value of $\epsilon = 10^{-3}$ to allow Gurobi to correctly distinguish between the two cases above.

6.5 Additional Experimental Results

Intercept Fig. 10 illustrates the return and worst-case error of Boolean-valued policies for Intercept. Only the policies with at least one case are optimal, with corresponding error of zero. As illustrated in the last plot in Fig. 11, the optimal policies fire whenever a threat is detected in the top right corner of the airspace.

Analysis of Problem Size Fig. 12, Fig. 13 and Fig. 14 summarize the total number of variables and constraints in the outer MIP formulations solved by CGPO at each iteration for Inventory, Reservoir and VTOL, respectively. Each iteration of constraint generation requires a roll-out from the dynamics and reward model, which in turn requires instantiate a new set of variables to hold the intermediate

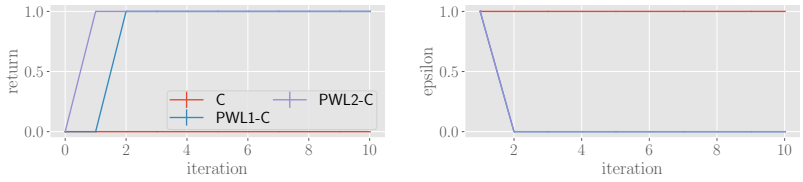


Fig. 10. Intercept: Simulated return (left) and optimal value of ε^* (right) as a function of the number of iterations of constraint generation.

$$fire = 1 \qquad fire = \begin{cases} 0 & \text{if } -0.136 \leq -0.2x - 0.02y \leq 0 \\ 1 & \text{otherwise} \end{cases} \qquad fire = \begin{cases} 0 & \text{if } -0.136 \leq -0.2x - 0.02y \leq 0 \\ 0 & \text{if } 0.001 \leq -x \leq 1 \\ 1 & \text{otherwise} \end{cases}$$

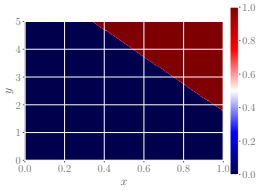


Fig. 11. Intercept: Examples of B, PWS1-B and PWS2-B policies (left to right) computed by CGPO, and visualization of PWS-B policy (bottom).

state and reward computations, and thus the number of variables and constraints grows linearly with the iterations. Even at the last iteration of CGPO, we see that the number of variables and constraints remains manageable.

6.6 Additional Related Work

The scheme of mixed discrete-continuous models can be traced back to the hybrid automata [19], and how to verify reachability and model checking [20]. In [17] a policy iteration approach has been taken to synthesize a feedback controller to a continuous-time system with discrete jumps. *Model predictive control* (MPC) approaches are especially appealing in the hybrid setting for controller synthesis [9], however they are not employed for worst-case analysis and policy optimization as in this work.

(Chance-) Constrained Optimization in Hybrid Systems The tool of constrained optimization is popular in policy optimization for hybrid systems [8]. For instance, [32] posed the problem of conjunctive synthesis and system falsification as constrained optimization. In the probabilistic or uncertain case, chance-constraints are popular for finding a robust policy; [6] used predictive control to synthesis the optimal controller in expectation under chance constraints. [27] utilized an recurrent neural network as parameterized policy for policy optimization under chance-constraints. Our work differs from the above as we optimize

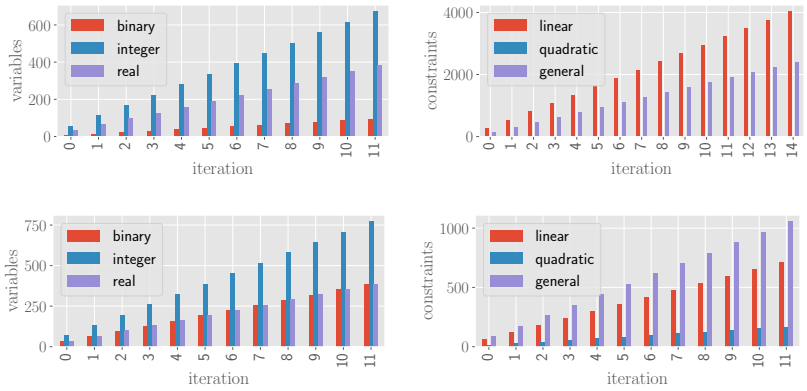


Fig. 12. Inventory: Number of decision variables (left) and constraints (right) for C policy (top) and PWS1-S policy (bottom) corresponding to the outer optimization problem at each iteration of CGPO.

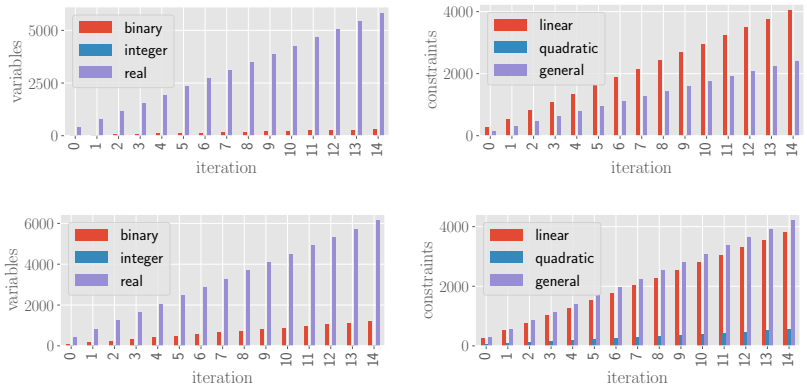


Fig. 13. Reservoir: Number of decision variables (left) and constraints (right) for C policy (top) and PWS1-S policy (bottom) corresponding to the outer optimization problem at each iteration of CGPO.

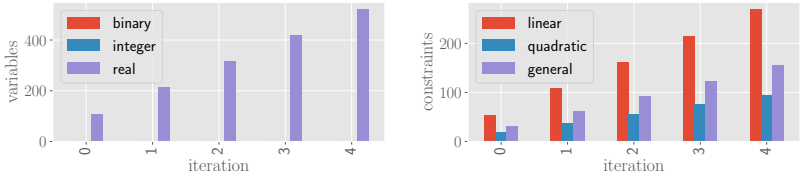


Fig. 14. VTOL: Number of decision variables (left) and constraints (right) for Q policy corresponding to the outer optimization problem at each iteration of CGPO.

a given policy structure, under the worst case, thus providing guarantees on the bounded policy optimality for the chosen policy class (also providing interpretability of the final optimized policy, due to the compactness of the policy class).

Constrained Policy Optimization in Reinforcement Learning A different stream of work incorporates safety constraints on parameterized policies [1,22,28]. However, they typically only guarantee convergence through gradient-based methods, and not bounded policy optimality as in our work. Furthermore, to accomplish this, they require differentiable – and often non-compact policy classes (i.e. such as neural network) – which makes them unsuitable for directly optimizing piecewise policy classes or other compact policy classes with discrete structure as studied in this work.

References

1. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: ICML. pp. 22–31. PMLR (2017)
2. Ahmed, A., Varakantham, P., Lowalekar, M., Adulyasak, Y., Jaillet, P.: Sampling based approaches for minimizing regret in uncertain markov decision processes (mdps). J. Artif. Int. Res. **59**(1), 229–264 (may 2017)
3. Albert, L.A.: A mixed-integer programming model for identifying intuitive ambulance dispatching policies. Journal of the Operational Research Society pp. 1–12 (2022)
4. Ariu, K., Fang, C., da Silva Arantes, M., Toledo, C., Williams, B.C.: Chance-constrained path planning with continuous time safety guarantees. In: AAAI Workshop (2017)
5. Åström, K.J.: Introduction to stochastic control theory. Courier Corporation (2012)
6. Blackmore, L., Ono, M., Bektassov, A., Williams, B.C.: A probabilistic particle-control approximation of chance-constrained stochastic predictive control. IEEE Transactions on Robotics **26**(3), 502–517 (2010). https://doi.org/10.1109/TRO.2010.2044948
7. Blankenship, J.W., Falk, J.E.: Infinitely constrained optimization problems. Journal of Optimization Theory and Applications **19**, 261–281 (1976)
8. Borrelli, F.: Constrained optimal control for hybrid systems. Springer (2003)
9. Borrelli, F., Bemporad, A., Morari, M.: Predictive control for linear and hybrid systems. Cambridge University Press (2017)
10. Bueno, T.P., de Barros, L.N., Mauá, D.D., Sanner, S.: Deep reactive policies for planning in stochastic nonlinear domains. In: AAAI. vol. 33, pp. 7530–7537 (2019)
11. Castro, P.M.: Tightening piecewise mccormick relaxations for bilinear problems. Computers & Chemical Engineering **72**, 300–311 (2015)
12. Chembu, A., Sanner, S., Khurram, H., Kumar, A.: Scalable and globally optimal generalized l1 k-center clustering via constraint generation in mixed integer linear programming. In: AAAI. vol. 37, pp. 7015–7023 (2023)
13. Corso, A., Moss, R., Koren, M., Lee, R., Kochenderfer, M.: A survey of algorithms for black-box safety validation of cyber-physical systems. Journal of Artificial Intelligence Research **72**, 377–428 (2021)

14. Dolgov, D., Durfee, E.: Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. p. 1326–1331. IJCAI'05, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
15. Farias, V.F., Van Roy, B.: Tetris: A Study of Randomized Constraint Sampling, pp. 189–201. Springer London, London (2006)
16. Farina, M., Giulioni, L., Scattolini, R.: Stochastic linear model predictive control with chance constraints–a review. Journal of Process Control **44**, 53–67 (2016)
17. Ferretti, R., Sassi, A., Zidani, H.: Recent advances in the numerical analysis of optimal hybrid control problems. HAL **2014** (2014)
18. Hauskrecht, M.: Approximate linear programming for solving hybrid factored mdps. In: International Symposium on Artificial Intelligence and Mathematics, AI&Math 2006, Fort Lauderdale, Florida, USA, January 4-6, 2006 (2006)
19. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292. IEEE (1996)
20. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Algorithmic analysis of nonlinear hybrid systems. IEEE transactions on automatic control **43**(4), 540–554 (1998)
21. Hickey, T., Ju, Q., Van Emden, M.H.: Interval arithmetic: From principles to implementation. Journal of the ACM **48**(5), 1038–1068 (2001)
22. Liu, T., Zhou, R., Kalathil, D., Kumar, P., Tian, C.: Policy optimization for constrained mdps with provable fast global convergence. arXiv preprint arXiv:2111.00552 (2021)
23. Low, S.M., Kumar, A., Sanner, S.: Sample-efficient iterative lower bound optimization of deep reactive policies for planning in continuous mdps. In: AAAI. vol. 36, pp. 9840–9848 (2022)
24. Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization **19**, 79–102 (2016)
25. Ono, M., Pavone, M., Kuwata, Y., Balaram, J.: Chance-constrained dynamic programming with application to risk-aware robotic space exploration. Autonomous Robots **39**, 555–571 (2015)
26. Patton, N., Jeong, J., Gimelfarb, M., Sanner, S.: A distributional framework for risk-sensitive end-to-end planning in continuous mdps. In: AAAI. vol. 36, pp. 9894–9901 (2022)
27. Petsagkourakis, P., Sandoval, I.O., Bradford, E., Galvanin, F., Zhang, D., del Rio-Chanona, E.A.: Chance constrained policy optimization for process control and optimization. Journal of Process Control **111**, 35–45 (2022)
28. Polosky, N., Da Silva, B.C., Fiterau, M., Jagannath, J.: Constrained offline policy optimization. In: ICML. pp. 17801–17810. PMLR (2022)
29. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons (2014)
30. Raghavan, A., Sanner, S., Khardon, R., Tadepalli, P., Fern, A.: Hindsight optimization for hybrid state and action mdps. In: AAAI. vol. 31 (2017)
31. Sanner, S.: Relational dynamic influence diagram language (rddl): Language description. Unpublished ms. Australian National University **32**, 27 (2010)
32. Sato, S., Waga, M., Hasuo, I.: Constrained optimization for hybrid system falsification and application to conjunctive synthesis. IFAC-PapersOnLine **54**(5), 217–222 (2021). https://doi.org/https://doi.org/10.1016/j.ifacol.2021.08.501, https://www.sciencedirect.com/science/article/pii/S2405896321012763, 7th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2021

33. Scala, E., Haslum, P., Thiébaux, S., Ramirez, M.: Interval-based relaxation for general numeric planning. In: ECAI, pp. 655–663. IOS Press (2016)
34. Scarf, H., Arrow, K., Karlin, S., Suppes, P.: The optimality of (s, s) policies in the dynamic inventory problem. Optimal pricing, inflation, and the cost of price adjustment pp. 49–56 (1960)
35. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. ArXiv **abs/1707.06347** (2017)
36. Schuurmans, D., Patrascu, R.: Direct value-approximation for factored MDPs. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems. vol. 14. MIT Press (2001)
37. Świechowski, M., Godlewski, K., Sawicki, B., Mańdziuk, J.: Monte carlo tree search: A review of recent modifications and applications. Artificial Intelligence Review **56**(3), 2497–2562 (2023)
38. Taitler, A., Gimelfarb, M., Gopalakrishnan, S., Mladenov, M., Liu, X., Sanner, S.: pyrddlgym: From rddl to gym environments. arXiv preprint arXiv:2211.05939 (2022)
39. Topin, N., Milani, S., Fang, F., Veloso, M.: Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. In: AAAI. vol. 35, pp. 9923–9931 (2021)
40. Vos, D., Verwer, S.: Optimal decision tree policies for markov decision processes. In: IJCAI. pp. 5457–5465 (2023)
41. Wang, Y., Wang, J., Zhang, W., Zhan, Y., Guo, S., Zheng, Q., Wang, X.: A survey on deploying mobile deep learning applications: A systemic and technical perspective. Digital Communications and Networks **8**(1), 1–17 (2022)
42. Wu, G., Say, B., Sanner, S.: Scalable planning with tensorflow for hybrid nonlinear domains. NeurIPS **30** (2017)