Appendix: 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, 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, Toronto, Canada
ssanner@mie.utoronto.ca

1 Appendix

In this supplement to the main paper, we provide further experimental details for successful implementation of CGPO. We also provide 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 the related work that was left out of the main text due to space limitations.

1.1 Linear Policy and 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.

1.2 The Nonlinear Intercept Domain

The intercept problem 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:

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

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.

1.3 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 [8]. 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.

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 [12]. To facilitate experimentation, we wrote a general-purpose routine for compiling RDDL code into a Gurobi mixed integer program formulation using the `pyRDDLGym` interface [14].

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:

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

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.

1.4 Additional Experimental Results

Intercept Fig. 1 illustrates the return and worst-case error of Boolean-valued policies for the intercept problem. Only the policies with at least one case are optimal, with corresponding error of zero. As illustrated in the last plot in Fig. 2, the optimal policies fire whenever a threat is detected in the top right corner of the airspace.

Analysis of Problem Size Fig. 3, Fig. 4 and Fig. 5 summarize the total number of variables and constraints in the outer MIP formulations solved by CGPO at each iteration for inventory, reservoir and VTOL control, 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 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.

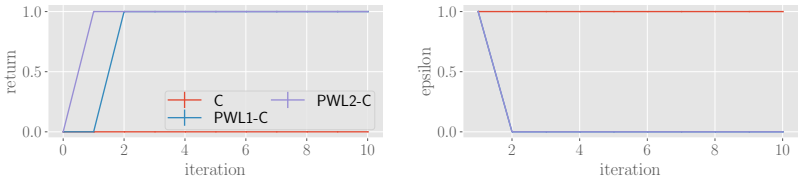

Fig. 1. Simulated return (left) and optimal value of ε^* (right) as a function of the number of iterations of constraint generation for the intercept domain.

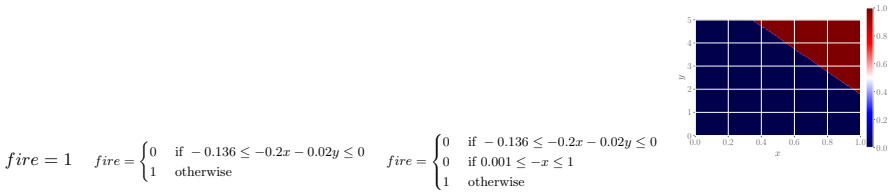

Fig. 2. From left to right, examples of learned B, PWS1-B and PWS2-B policies and visualization of PWS-B policy for the intercept problem.

1.5 Additional Related Work

The scheme of mixed discrete-continuous models can be traced back to the hybrid automata [6], and how to verify reachability and model checking [7]. In [5] a policy iteration to 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 [4], 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 [3]. For instance, [13] 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; [2] used predictive control

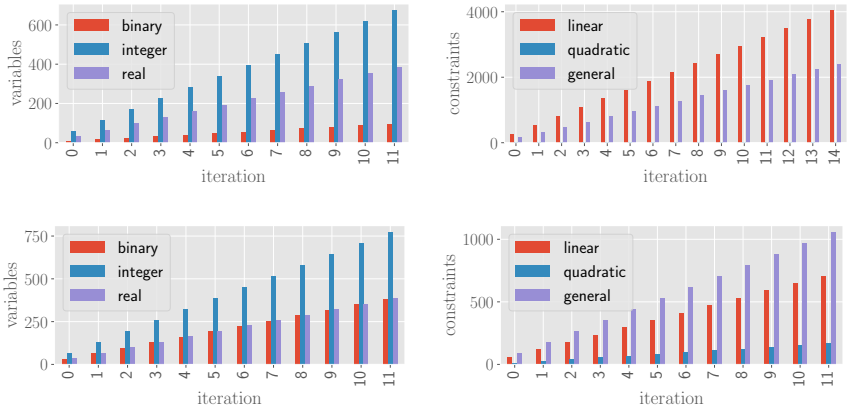

Fig. 3. Number of decision variables (left) and constraints (right) for C policy (top row) and PWS1-S policy (bottom row) corresponding to the outer optimization problem at each iteration applied to the inventory problem.

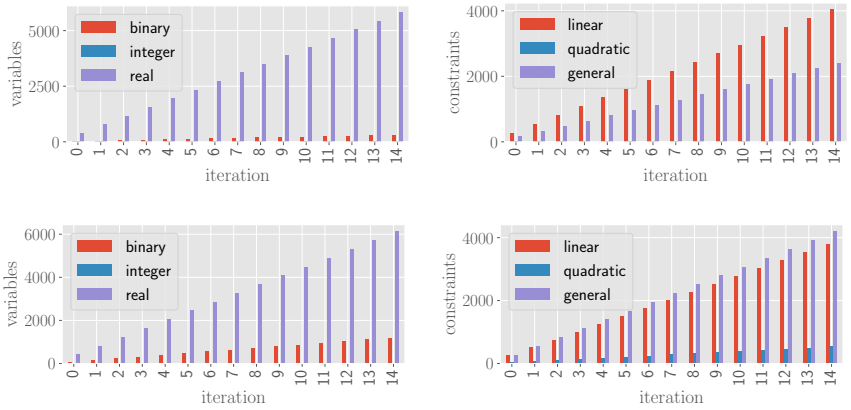

Fig. 4. Number of decision variables (left) and constraints (right) for C policy (top row) and PWS1-S policy (bottom row) corresponding to the outer optimization problem at each iteration applied to the reservoir problem.

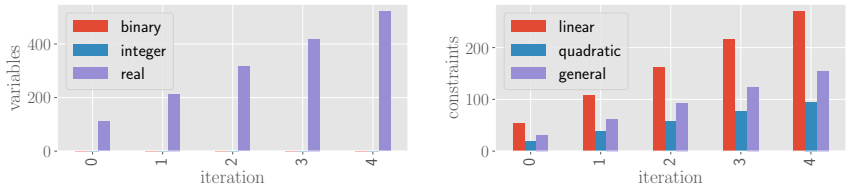

Fig. 5. Number of decision variables (left) and constraints (right) for Q policy corresponding to the outer optimization problem at each iteration applied to the VTOL problem.

to synthesis the optimal controller in expectation under chance constraints. [10] utilized an recurrent neural network as parameterized policy for policy optimization under chance-constraints. Our work differs from the above as we optimize 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 [1,9,11]. However, they typically only guarantee convergence through (policy) 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. 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
3. Borrelli, F.: Constrained optimal control for hybrid systems. Springer (2003)
4. Borrelli, F., Bemporad, A., Morari, M.: Predictive control for linear and hybrid systems. Cambridge University Press (2017)
5. Ferretti, R., Sassi, A., Zidani, H.: Recent advances in the numerical analysis of optimal hybrid control problems. HAL **2014** (2014)
6. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292. IEEE (1996)
7. 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)

8. Hickey, T., Ju, Q., Van Emden, M.H.: Interval arithmetic: From principles to implementation. Journal of the ACM **48**(5), 1038–1068 (2001)
9. 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)
10. 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)
11. Polosky, N., Da Silva, B.C., Fiterau, M., Jagannath, J.: Constrained offline policy optimization. In: ICML. pp. 17801–17810. PMLR (2022)
12. Sanner, S.: Relational dynamic influence diagram language (rddl): Language description. Unpublished ms. Australian National University **32**, 27 (2010)
13. 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
14. Taitler, A., Gimelfarb, M., Gopalakrishnan, S., Mladenov, M., Liu, X., Sanner, S.: pyrddlgym: From rddl to gym environments. arXiv preprint arXiv:2211.05939 (2022)