Constrained Policy Optimization

Joshua Achiam, David Held, Aviv Tamar, Pieter Abbeel

Thanh Cuong Le 407996
Paul Hasenbusch 398475

TU Berlin

November 25, 2022

Outline

- Introduction
- Theoretical foundations
  - CMDP
  - Constraint policy optimization
  - Trust Region Method
  - CPO with trust region optimization
- CPO Algorithm
- Experiments

Introduction

- Scenario: A robot that is supposed to get out of an alley without hitting the houses (Humanoid - Circle)
- Things to concern: Efficiency and Safeness
- Problem: The robot could learn bad behaviour and can wreck havoc
- Solution: We introduce constraints in the RL training process of the agent
- Goal: Iterative improvement and satistfaction of safety constraints
- How does it compare to fix penalties?

Markov decision process (MDP)

We define our Markov decision process as a set of tuple (S, A, R, P, μ) whereas:

S : Set of states, A : Set of actions

$R : S$ x A x $S \longrightarrow \mathbb{R}$ the reward function

$P : S$ x A x $S \longrightarrow [0, 1]$ the transition probability function

$\mu : S \longrightarrow [0, 1]$ the distribution of starting state

also, we construct $\pi : S \longrightarrow \mathbf{P}(A)$ as the stationary policy

Thus, the performance measure is defined as the following function:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right] .$$

Markov decision process (MDP)

We define our Markov decision process as a set of tuple (S, A, R, P, μ) whereas:

S : Set of states, A : Set of actions

$R : S \times A \times S \longrightarrow \mathbb{R}$ the reward function

$P : S \times A \times S \longrightarrow [0, 1]$ the transition probability function

$\mu : S \longrightarrow [0, 1]$ the distribution of starting state

also, we construct $\pi : S \longrightarrow \mathbf{P}(A)$ as the stationary policy

Thus, the performance measure is defined as the following function:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right] .$$

In this scenario, we express the difference between performance measures of different policies for $A^\pi(s, a)$ - an advantage function, as following:

$$J(\pi') - J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^{\pi'}} [A^\pi(s, a)]$$

Constrained Markov decision process (CMDP)

A constrained Markov decision process (CMDP) is an Markov decision process (MDP) (S, A, R, P, μ) augmented with a set C of auxiliary cost functions $C_1, ..., C_m : S \times A \times S \to \mathbb{R}$.

For $i = 1, ..., m$ we define the expected discounted return of policy π with respect to cost function C_i by

$$J_{C_i} : \Pi \to \mathbb{R} \ , \quad J_{C_i}(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1})\right] \ .$$

Given limits $d_1, ..., d_m \in \mathbb{R}$ the set of feasible stationary policies for the CMDP is given by

$$\Pi_C = \{\pi \ : \ J_{C_i}(\pi) \leq d_i \text{ for all } i = 1, ..., m\} \ .$$

Constraint policy optimization

Let $\Pi_\theta \subset \Pi$ be the set of parametrized policies under parameter θ, on which we want to conduct our policy search algorithm to avoid the curse of dimensionality.

CPO's idea: We optimize over $\Pi_\theta \bigcap \Pi_C$. For $\delta > 0$ step size, D as a distance measure

$$\pi_{k+1} = \arg\max_{\pi \in \Pi_\theta} J(\pi),$$

s.t

$$J_{C_i}(\pi) \leq d_i, \quad i = 1, ..m$$

and

$$D(\pi, \pi_k) \leq \delta$$

Trust region methods

We optimize the reward function by maximizing the reward advantage function $A^{\pi}_{R}(s,a)$ subject to a Kullback-Leibler (KL) divergence constraint D_{KL}.

$$\pi_{k+1} = \arg\max_{\pi \in \Pi_\theta} \mathbb{E}_{s \sim d^{\pi^k}} \left[A^{\pi}_{R}(s,a)\right]$$

such that

$$\mathbb{E}_{s \sim d^{\pi^k}}[D_{KL}(\pi||\pi_k)[s]] \leq \delta,$$

in which the trust region is defined as the set

$$\{\pi_\theta \in \Pi_\theta : \mathbb{E}_{s \sim d^{\pi^k}}[D_{KL}(\pi||\pi_k)[s]] \leq \delta.\}$$

Trust region methods

We optimize the reward function by maximizing the reward advantage function $A_R^\pi(s, a)$ subject to a Kullback-Leibler (KL) divergence constraint D_{KL}.

$$\pi_{k+1} = \arg\max_{\pi \in \Pi_\theta} \mathbb{E}_{s \sim d^{\pi^k}} \left[A_R^\pi(s, a) \right]$$

such that

$$\mathbb{E}_{s \sim d^{\pi^k}} [D_{KL}(\pi || \pi_k)[s]] \leq \delta,$$

in which the trust region is defined as the set

$$\{\pi_\theta \in \Pi_\theta : \mathbb{E}_{s \sim d^{\pi^k}} [D_{KL}(\pi || \pi_k)[s]] \leq \delta.\}$$

Question: What are the pros of applying a trust region method in our optimization of neural network policies?

Trust region update performance

With π_{k+1} and π_k as above, the difference between the policy performance of π_{k+1} and π_k could be bounded below as

$$J(\pi_{k+1}) - J(\pi_k) \geq \frac{-\sqrt{2\delta}\gamma\epsilon^{\pi_{k+1}}}{(1-\gamma)^2}$$

whereas $\epsilon^{\pi_{k+1}} = \max_s \mathbb{E}_{a\sim\pi_{k+1}}[A^{\pi}(s,a)]$

Trust region update performance

With π_{k+1} and π_k as above, the difference between the policy performance of π_{k+1} and π_k could be bounded below as

$$J(\pi_{k+1}) - J(\pi_k) \geq \frac{-\sqrt{2\delta}\gamma\epsilon^{\pi_{k+1}}}{(1-\gamma)^2}$$

whereas $\epsilon^{\pi_{k+1}} = \max_s \mathbb{E}_{a\sim\pi_{k+1}}[A^\pi(s,a)]$

Knowing this result, each update guarantees a monotonic improvement of the performance measure.

$\Rightarrow$ In case we construct a CPO model based on a trust region method, our training process with have this property.

Trust Region Method in general[1]

Given a function $f : D \to \mathbb{R}$, and a starting point $x_0 \in D$ do for $k = 1, 2, ...$

[1] Taken from [2].

Trust Region Method in general[1]

Given a function $f : D \to \mathbb{R}$, and a starting point $x_0 \in D$ do for $k = 1, 2, ...$

1. During each iteration choose a local model f_k of the objective function f
   - $f_k(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + ...$

[1] Taken from [2].

Trust Region Method in general[1]

Given a function $f : D \to \mathbb{R}$, and a starting point $x_0 \in D$ do for $k = 1, 2, \ldots$

1. During each iteration choose a local model f_k of the objective function f
   - $f_k(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \ldots$
2. Choose $\delta_k > 0$ and define $B(x_k, \delta_k)$
   - $B(x_k, \delta_k) = \{x \in D : d(x_k, x) \leq \delta_k\}$, where d is some metric on D

[1] Taken from [2].

Trust Region Method in general[1]

Given a function $f : D \to \mathbb{R}$, and a starting point $x_0 \in D$ do for $k = 1, 2, ...$

1. During each iteration choose a local model f_k of the objective function f
   - $f_k(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + ...$
2. Choose $\delta_k > 0$ and define $B(x_k, \delta_k)$
   - $B(x_k, \delta_k) = \{x \in D : d(x_k, x) \leq \delta_k\}$, where d is some metric on D
3. compute new search direction d_k by solving

$$\min_{d \in B(x_k, \delta_k)} f_k(d)$$

[1] Taken from [2].

Trust region optimization for constraint MDPs

By applying the trust region methods with suitable coefficients instead of penalties on policy divergence, we can guarantee a monotonic improvement while being able to take large steps.
In each iteration we update:

$$\pi_{k+1} = \underset{\pi \in \Pi_\theta}{\arg\max}\, \mathbb{E}_{s \sim d^{\pi^k}} \left[A_R^{\pi}(s, a) \right]$$

with constraints:

$$J_{C_i}(\pi_k) + \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi^k}} \left[A_R^{\pi}(s, a) \right] \leq d_i$$

and

$$\mathbb{E}_{s \sim d^{\pi^k}} [D_{KL}(\pi || \pi_k)[s]] \leq \delta,$$

However, it might be possible for an update to violate our constraint. Therefore we come to the question: How bad is this violation?

Trust region optimization for constraint MDPs

CPO Update Worst-Case Violation

Let π_k, $\pi_{k+1} \in \Pi_\theta$, the upper bound on the performance measure regarding constraint set C_i is:

$$J_{C_i}(\pi_{k+1}) \leq d_i + \frac{-\sqrt{2\delta}\gamma\epsilon^{\pi_{k+1}}}{(1-\gamma)^2}$$

whereas $\epsilon_{C_i}^{\pi_{k+1}} = \max_s \mathbb{E}_{a \sim \pi_{k+1}}[A_{C_i}^{\pi}(s, a)]$

Trust region optimization for constraint MDPs

CPO Update Worst-Case Violation

Let π_k, $\pi_{k+1} \in \Pi_\theta$, the upper bound on the performance measure regarding constraint set C_i is:

$$J_{C_i}(\pi_{k+1}) \leq d_i + \frac{-\sqrt{2\delta}\gamma\epsilon^{\pi_{k+1}}}{(1-\gamma)^2}$$

whereas $\epsilon_{C_i}^{\pi_{k+1}} = \max_s \mathbb{E}_{a\sim\pi_{k+1}}[A_{C_i}^{\pi}(s,a)]$

The upper inequality tells us about the worst case of a constraint violating update.

Approximately Solving the CPO Update

Now we turn to parametrized policies π_θ

Approximately Solving the CPO Update

Now we turn to parametrized policies π_θ

$$\theta_{k+1} = \arg\max_{\theta} \ \nabla J(\theta_k)^\top (\theta - \theta_k)$$

$$\text{s.t. } J_{C_i}(\theta_k) + \nabla J_{C_i}(\theta_k)^\top (\theta - \theta_k) \leq d_i \text{ for } i = 1, ..., m$$
$$\frac{1}{2}(\theta - \theta_k)^\top H(\theta - \theta_k) \leq \delta$$

Where H denotes the Hessian of the KL-divergence, we assume it to be positive definite

Approximately Solving the CPO Update

Now we turn to parametrized policies π_θ

$$\theta_{k+1} = \arg\max_{\theta} \ \nabla J(\theta_k)^\top (\theta - \theta_k)$$

$$\begin{aligned} \text{s.t. } & J_{C_i}(\theta_k) + \nabla J_{C_i}(\theta_k)^\top (\theta - \theta_k) \leq d_i \text{ for } i = 1, ..., m \\ & \frac{1}{2}(\theta - \theta_k)^\top H(\theta - \theta_k) \leq \delta \end{aligned}$$

Where H denotes the Hessian of the KL-divergence, we assume it to be positive definite

Note both the KL-divergence itself and its gradient are zero when plugging in the same distributions

Approximately Solving the CPO Update

Now we turn to parametrized policies π_θ

$$\theta_{k+1} = \arg\max_{\theta} \ \nabla J(\theta_k)^\top (\theta - \theta_k)$$

$$\begin{aligned} \text{s.t.} \ & J_{C_i}(\theta_k) + \nabla J_{C_i}(\theta_k)^\top (\theta - \theta_k) \leq d_i \text{ for } i = 1, \ldots, m \\ & \frac{1}{2}(\theta - \theta_k)^\top H (\theta - \theta_k) \leq \delta \end{aligned}$$

Where H denotes the Hessian of the KL-divergence, we assume it to be positive definite

Note both the KL-divergence itself and its gradient are zero when plugging in the same distributions

$\longrightarrow$ can be computed efficiently by solving the dual problem

Feasibility

In case θ_k becomes unfeasible a recovery method is necessary.

Feasibility

In case θ_k becomes unfeasible a recovery method is necessary.
$\longrightarrow$ might be due to approximation errors

Feasibility

In case θ_k becomes unfeasible a recovery method is necessary.
$\longrightarrow$ might be due to approximation errors
In case there is only one constraint a recovery method is given by

$$\theta_{k+1} = \theta_k - \sqrt{\frac{2\delta}{b^\top H^{-1} b}} H^{-1} b$$

where $b := \nabla J_C(\theta_k)$.

Feasibility

In case θ_k becomes unfeasible a recovery method is necessary.
$\longrightarrow$ might be due to approximation errors
In case there is only one constraint a recovery method is given by

$$\theta_{k+1} = \theta_k - \sqrt{\frac{2\delta}{b^\top H^{-1} b}} H^{-1} b$$

where $b := \nabla J_C(\theta_k)$. This decreases the constraint value

$$J_C(\theta_k) + b^\top(\theta_{k+1} - \theta_k) = J_C(\theta_k) - \underbrace{\sqrt{\frac{2\delta}{b^\top H^{-1} b}} b^\top H^{-1} b}_{>0} < J_C(\theta_k) \ .$$

Note: A positive definite $\iff A^{-1}$ positive definite.

Feasibility

In case θ_k becomes unfeasible a recovery method is necessary.
$\longrightarrow$ might be due to approximation errors
In case there is only one constraint a recovery method is given by

$$\theta_{k+1} = \theta_k - \sqrt{\frac{2\delta}{b^\top H^{-1} b}} H^{-1} b$$

where $b := \nabla J_C(\theta_k)$. This decreases the constraint value

$$J_C(\theta_k) + b^\top(\theta_{k+1} - \theta_k) = J_C(\theta_k) - \underbrace{\sqrt{\frac{2\delta}{b^\top H^{-1} b}} b^\top H^{-1} b}_{>0} < J_C(\theta_k) \ .$$

Note: A positive definite $\iff A^{-1}$ positive definite.
Moreover one can easily check that $\frac{1}{2}(\theta_{k+1} - \theta_k)^\top H(\theta_{k+1} - \theta_k) = \delta$.

Constrained Policy Optimization

Input Initial policy $\pi_0 \in \Pi_\theta$

for $k = 0, 1, 2, ...$ **do**

Sample a set of trajectories $\mathcal{D} = \{\tau\} \sim \pi_k = \pi(\theta_k)$

Form estimates of the objective function and constraints with $\mathcal{D}$

if approximate CPO is feasible **then**

Solve dual problem and compute policy proposal θ^*

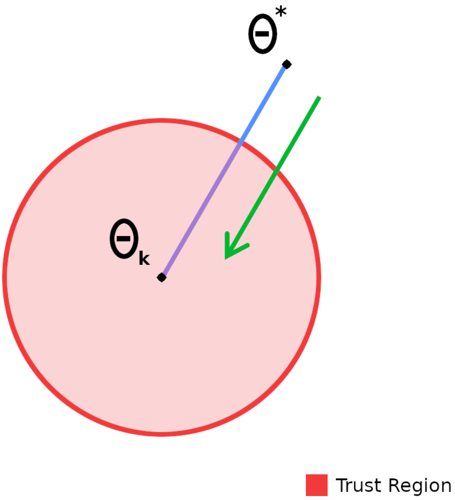
else

Compute recovery policy proposal θ^*

end if

Obtain θ_{k+1} by backtracking linesearch to enforce satisfaction of sample estimates of constraints

end for



2

[2] Own creation .

Experiments, Key Questions

Experiments, Key Questions

- Does CPO succeed at enforcing behavioral constraints?

Experiments, Key Questions

- Does CPO succeed at enforcing behavioral constraints?
- What benefits are conferred by using constraints instead of fixed penalties?

Experiments, Key Questions

- Does CPO succeed at enforcing behavioral constraints?
- What benefits are conferred by using constraints instead of fixed penalties?
- How much does it help to constrain a cost upper bound, instead of directly constraining the cost?

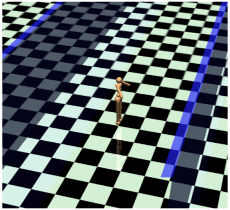
Experiment Setting

Agents

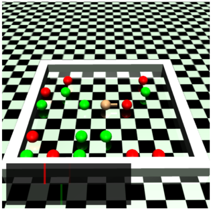
- point-mass ($S \subseteq \mathbb{R}^9, A \subseteq \mathbb{R}^2$)
- quadruped robot ($S \subseteq \mathbb{R}^{32}, A \subseteq \mathbb{R}^8$)
- humanoid ($S \subseteq \mathbb{R}^{102}, A \subseteq \mathbb{R}^{10}$)

Tasks

- Gather
- Circle



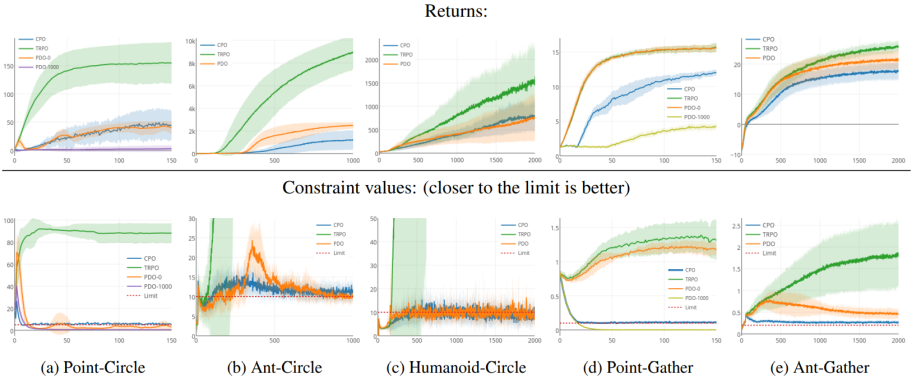
(a) Humanoid-Circle



(b) Point-Gather [3]

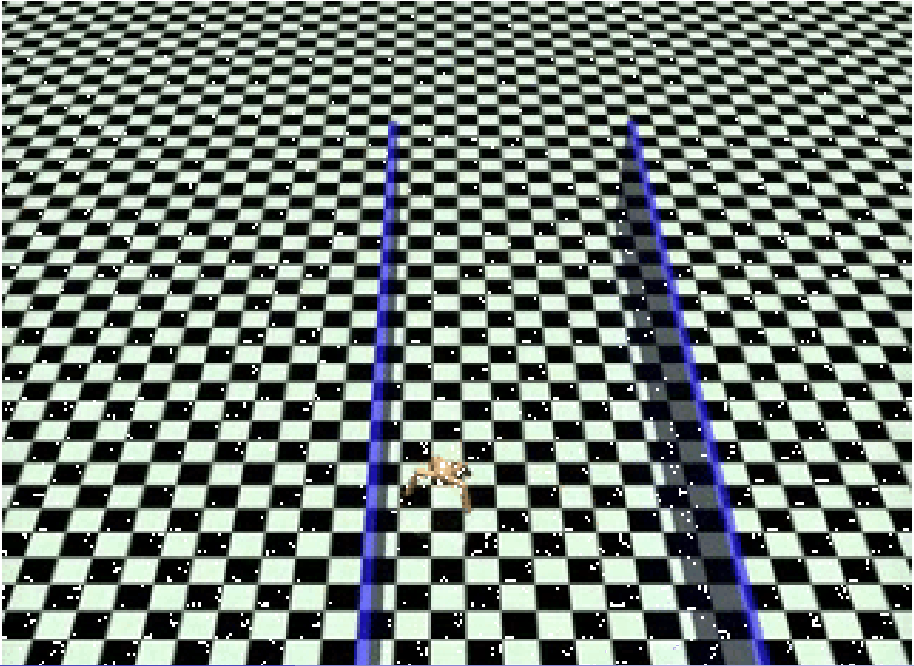
[3] Taken from [1, Figure 2] .

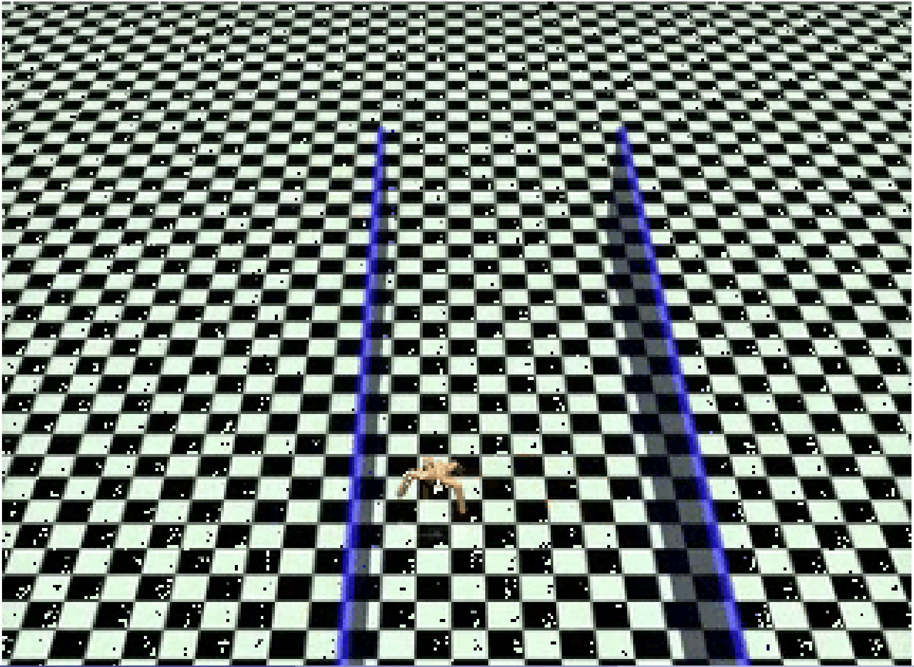
Does CPO succeed at enforcing behavioral constraints?

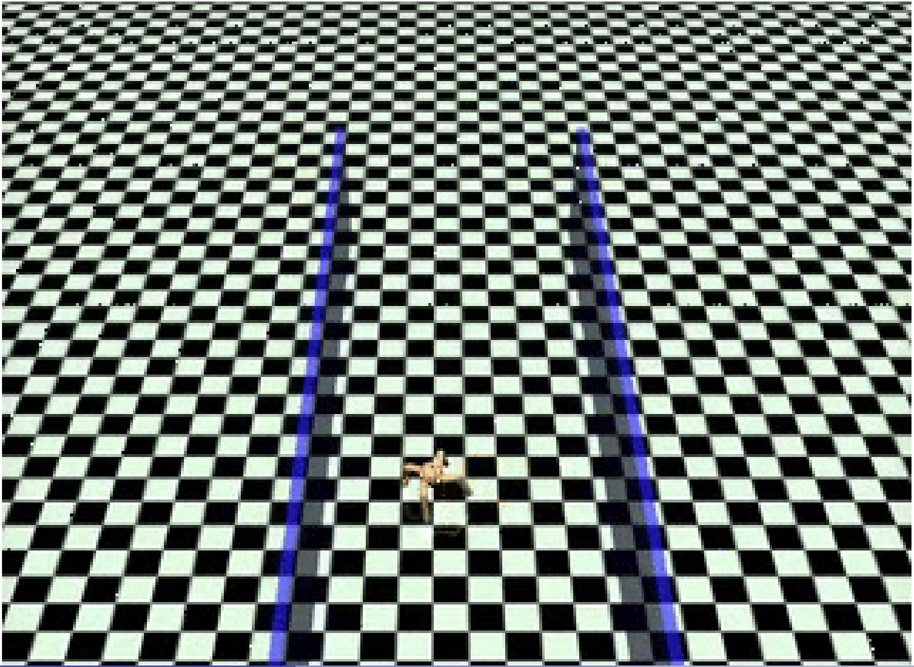


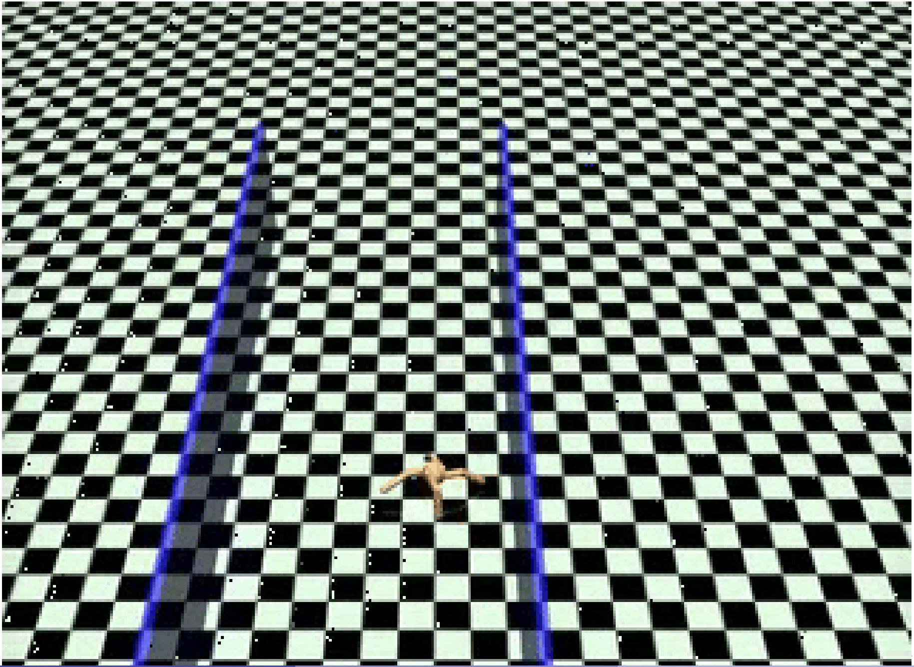
[4]

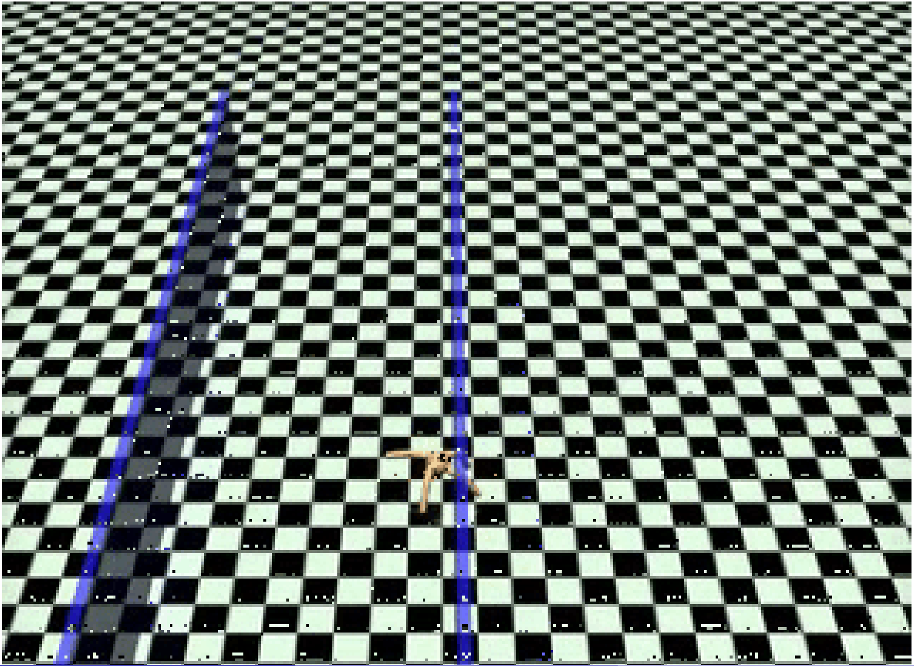
[4] Taken from [1, Figure 1] .

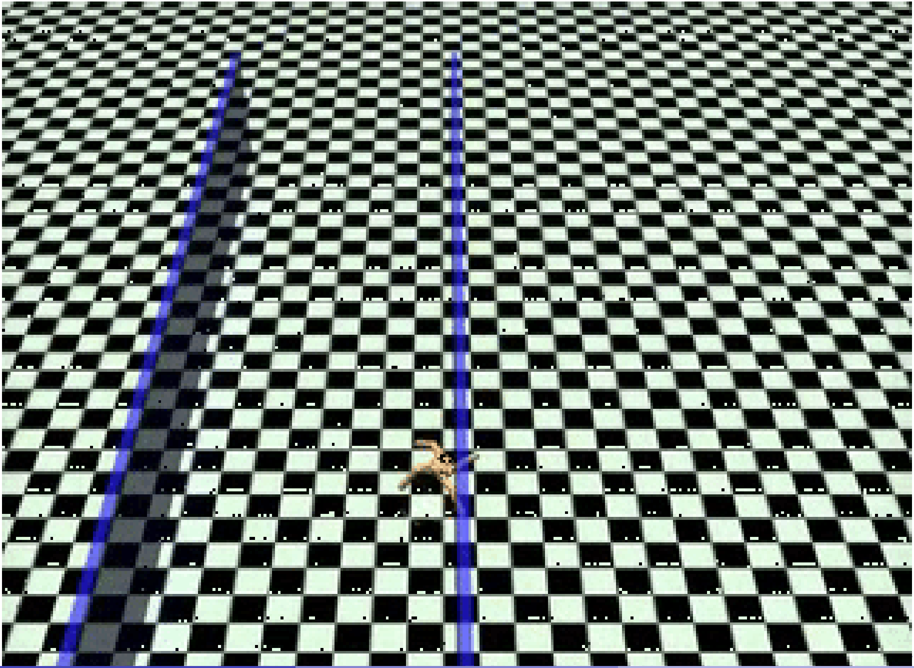


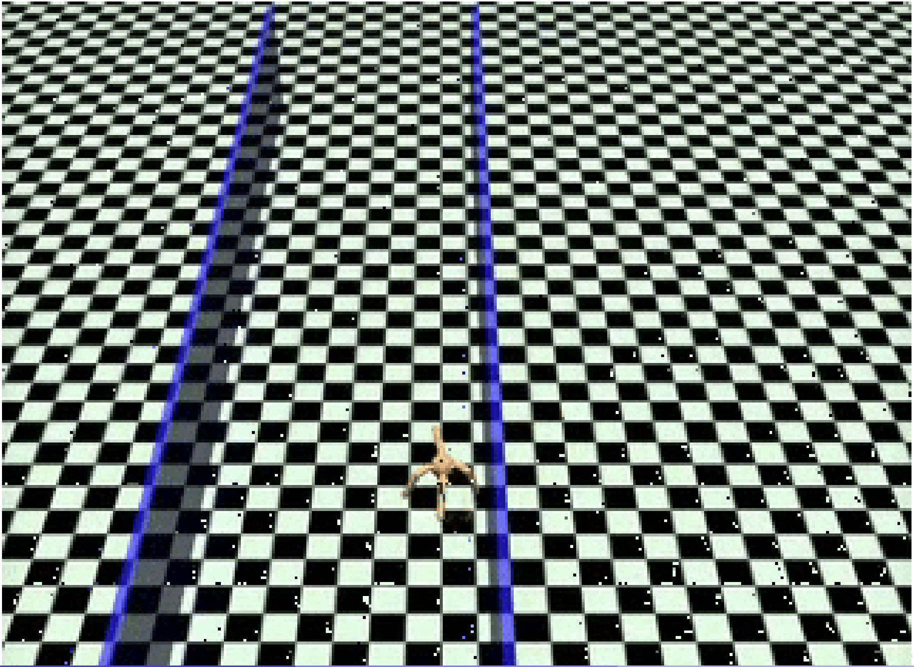


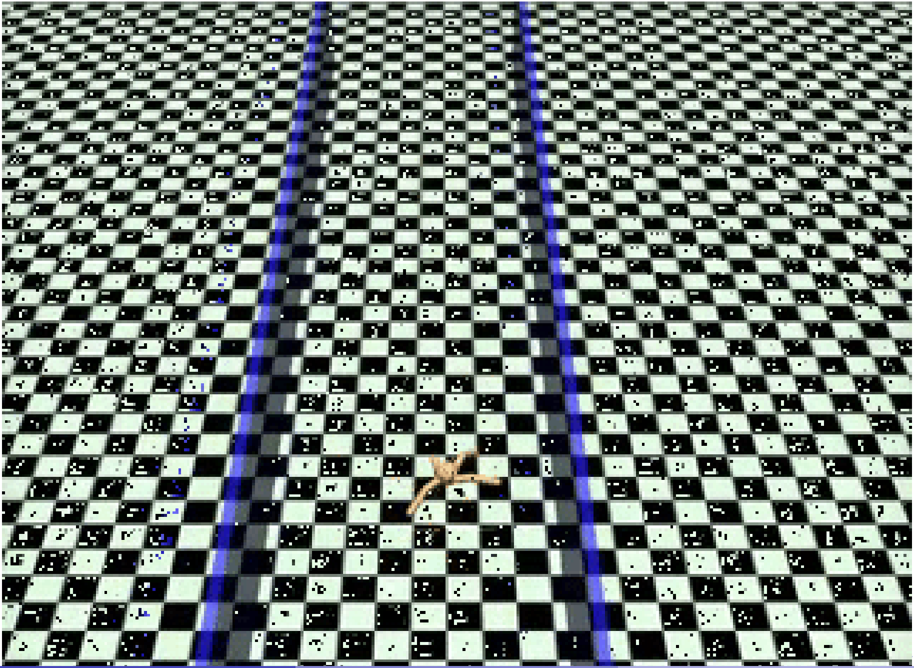


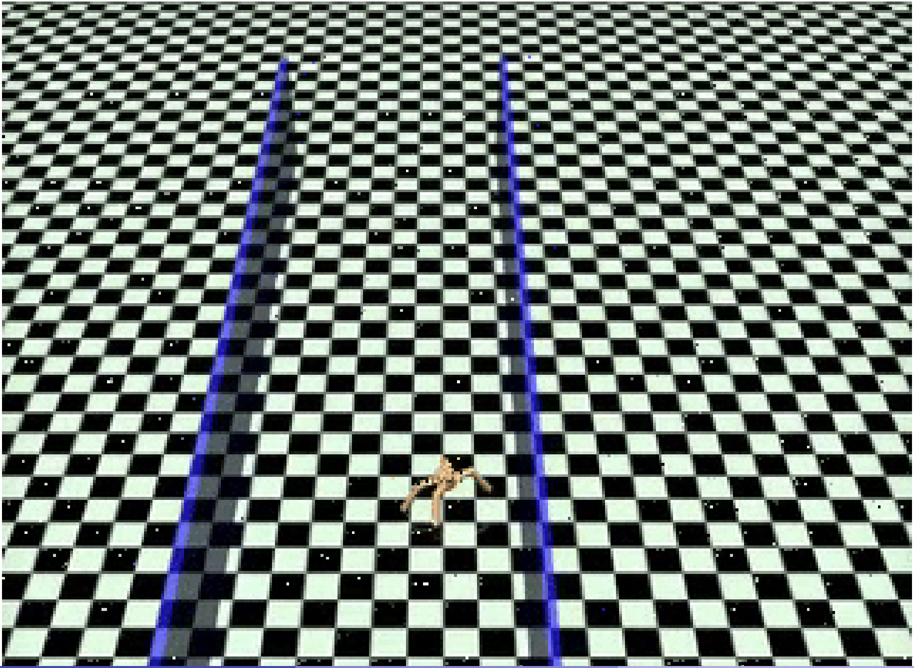


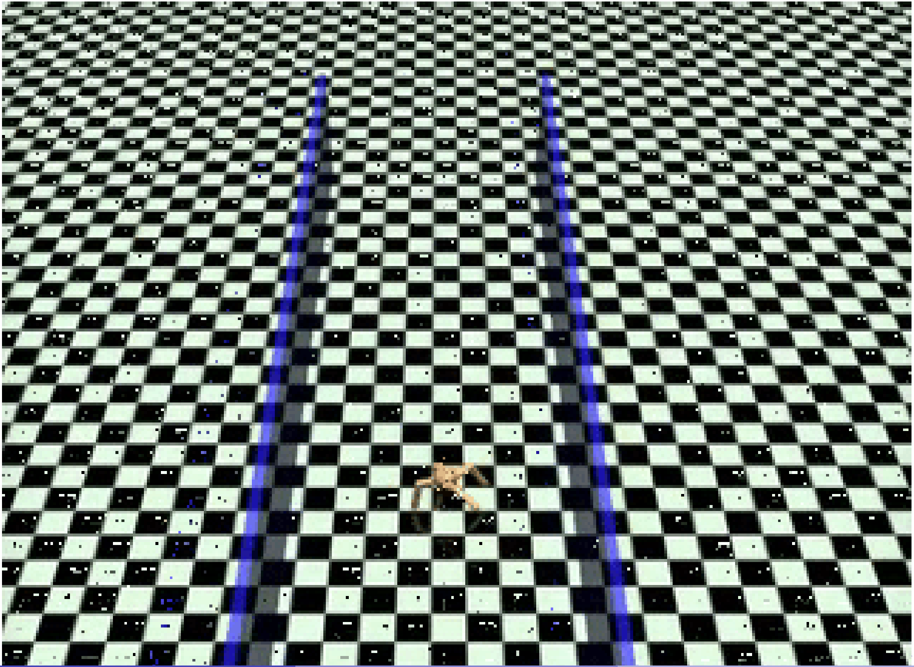


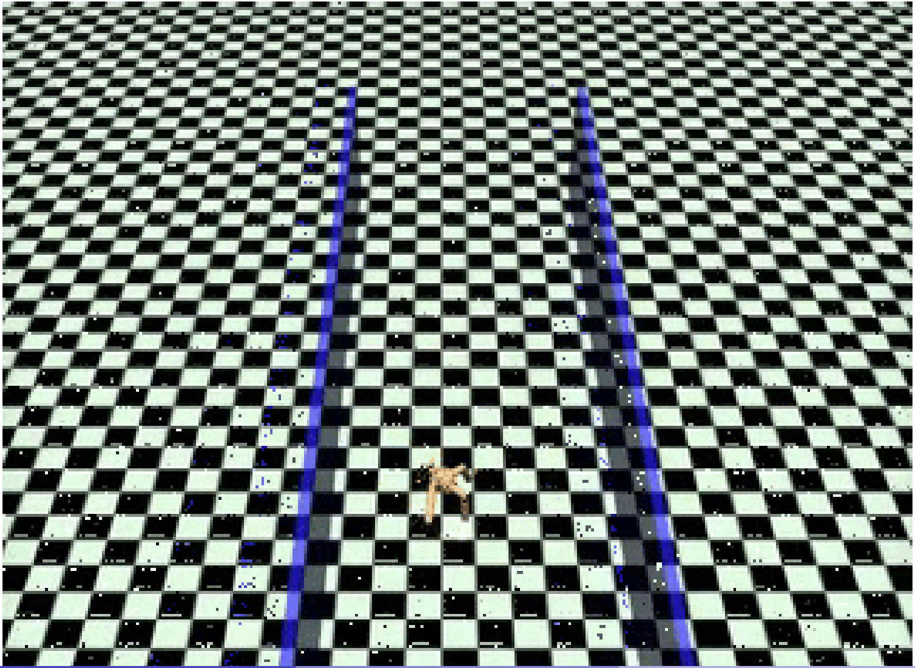


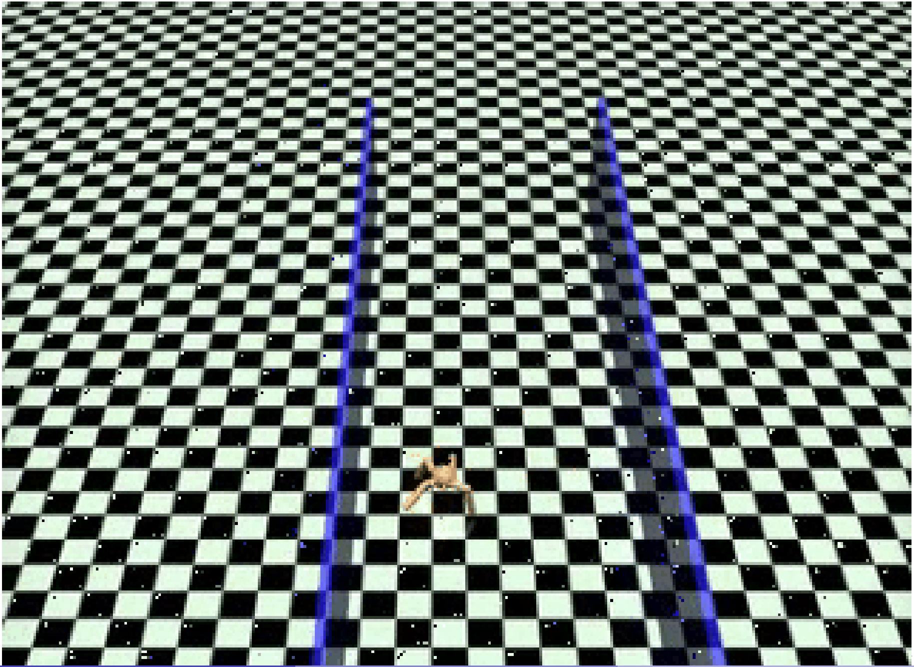


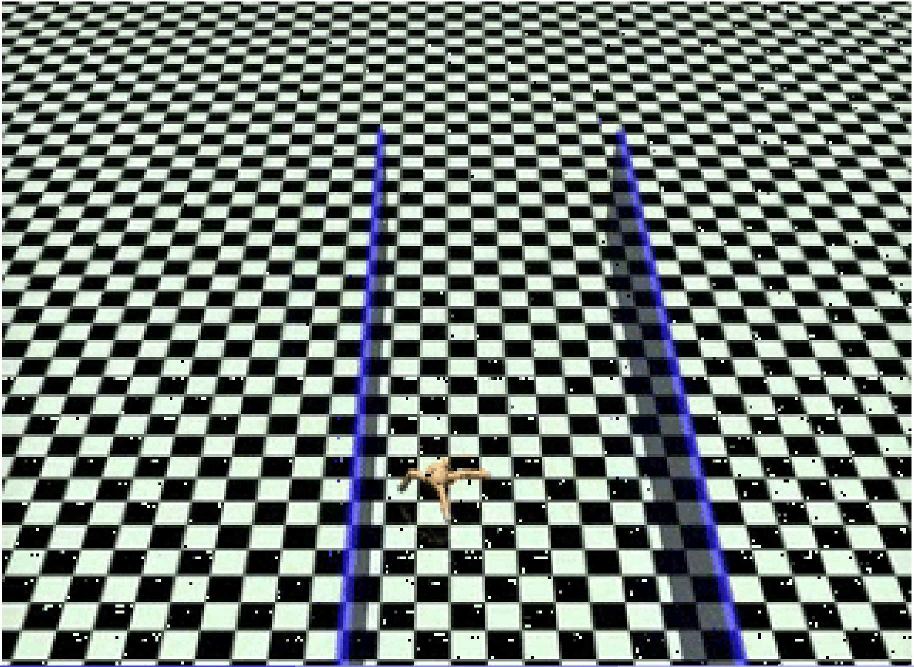




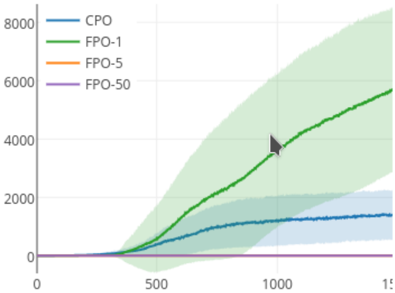


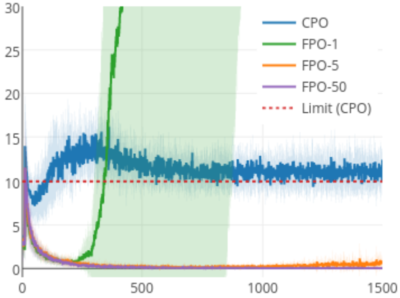






What benefits are conferred by using constraints instead of fixed penalties?





[5]

[5] Taken from [1, Figure 4] .

Cost shaping

Constrain an upper bound on the original constraint

$$C_i^+(s,a,s') = C_i(s,a,s') + \Delta_i(s,a,s') \ ,$$

where Δ_i correlates in some useful way with C_i.

Cost shaping

Constrain an upper bound on the original constraint

$$C_i^+(s, a, s') = C_i(s, a, s') + \Delta_i(s, a, s') \ ,$$

where Δ_i correlates in some useful way with C_i.
$\longrightarrow$ minimize the risk that due to the many approximations the original constraints are violated

Cost shaping

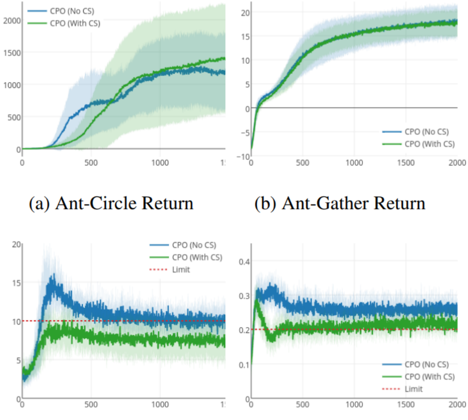
Constrain an upper bound on the original constraint

$$C_i^+(s,a,s') = C_i(s,a,s') + \Delta_i(s,a,s') \ ,$$

where Δ_i correlates in some useful way with C_i.
$\longrightarrow$ minimize the risk that due to the many approximations the original constraints are violated

Alchiam et al. [1] partitioned states into safe and unsafe and chose Δ to be the probability to enter an unsafe state within a fixed time horizon, according to a model which is simultaneously trained.

How much does it help to constrain a cost upper bound?



(a) Ant-Circle Return (b) Ant-Gather Return

(c) Ant-Circle C Return (d) Ant-Gather C Return [6]

[6] Taken from [1, Figure 3] .

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel.
Constrained policy optimization.
In *International conference on machine learning*, pages 22–31, Sydney, Australia, 2017. PMLR.

D. Hömberg.
Nichtlineare Optimierung, SoSe 21.
University Lecture, 2021.

GIF on page 17 taken from:
https://bair.berkeley.edu/blog/2017/07/06/cpo/