Safe Exploration in Continuous Action Spaces

THE BACKPROPERS
Divyani Gaur, Rajarshi Dutta, Udvas Basak

The Problem

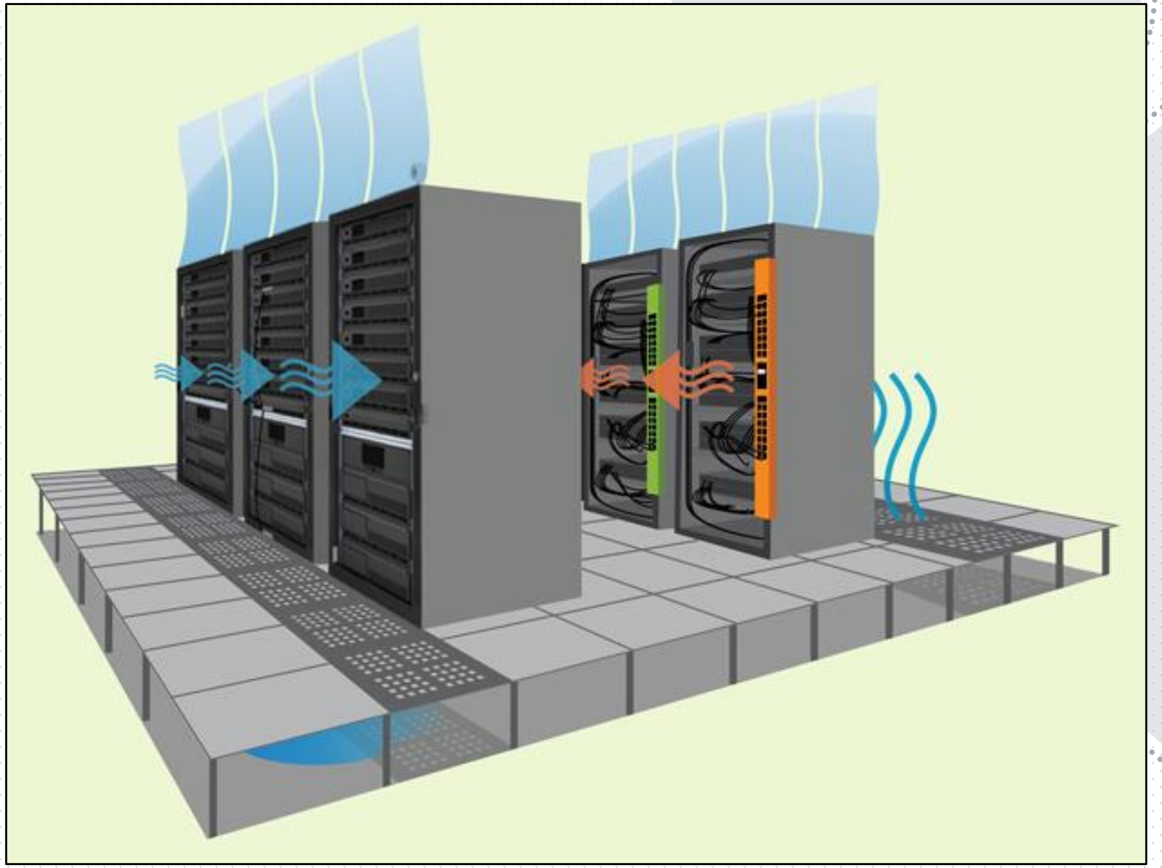
Data Center Cooling

- A data center needs to be cooled appropriately, ensuring minimal power consumption

- **Summarized Dynamics**:
  - CPUs/GPUs radiate heat, which are redirected to the outside(Heat Source)
  - Through circulation and convection, heat diffuses into the whole center
  - Coolers and/or radiators dispose the heat outside the center(Heat Sink)

- **Possible Constraints**:
  - Humans may need to interact with these centers
  - Melting point of the components involved
  - Operating conditions of the radiators/coolers involved

- Second-order systems, controlled by first order laws of cooling, and second-order mass flow equations.



https://www.raritan.com/ap/blog/detail/types-of-data-center-cooling-techniques

Robotic Arms

- Controlling a robotic arm is a widely used RL problem
- **Summarized Dynamics**:
  - Translation/Rotation at the root of the arm through forces and torques(servos)
  - Rotation at each arm(servos)
- **Possible Constraints**:
  - Mechanical limits in rotation
  - Mechanical limits in angular velocity/acceleration
- Second-order system, governed by Newton's laws of motion

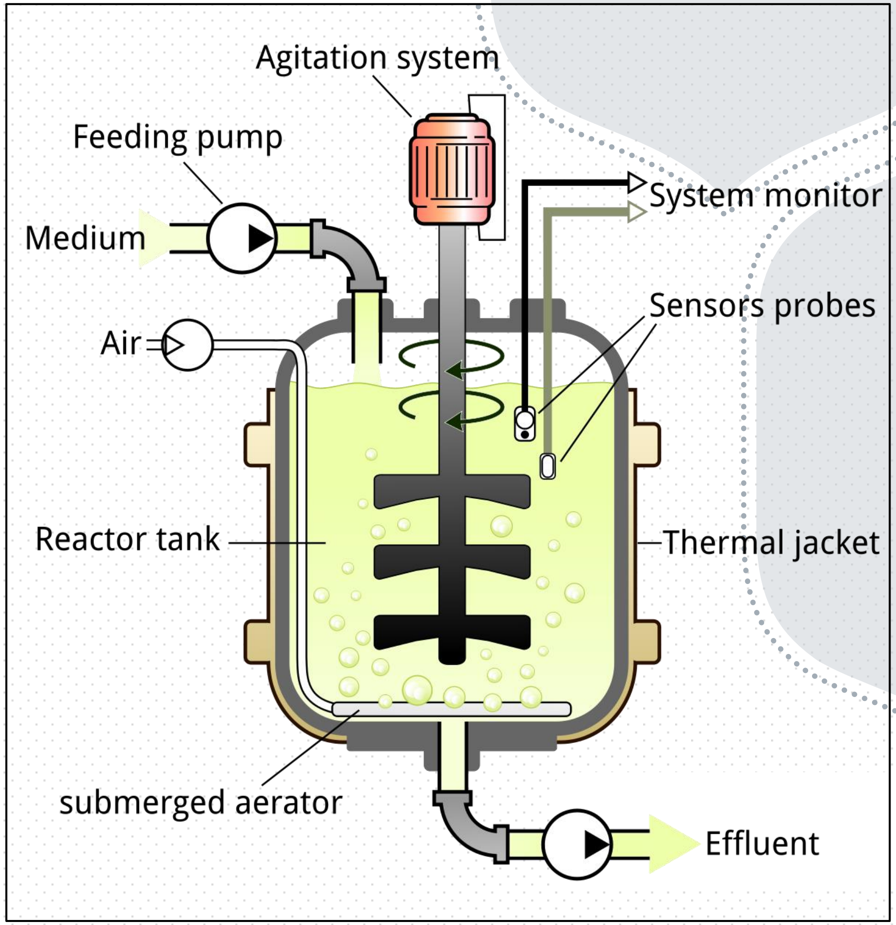
Bioreactor Control

- Conduct biochemical reactions, leading to formation of biomass via an exothermic reaction

- **Summarized Dynamics**:
  - Transport of substances (oxygen, nutrients) into the bioreactor
  - Rate of product formation like biomass, metabolites etc.
  - Certain inhibitors/chemicals released which might affect the growth rate

- **Possible Constraints**:
  - The bioreactor controller must operate within a pressure range.
  - Capacity limitations for the bioreactor components involved
  - An optimal temperature range for the chemical reactions to proceed

- First Order Differential Equation containing concentration and temperature terms



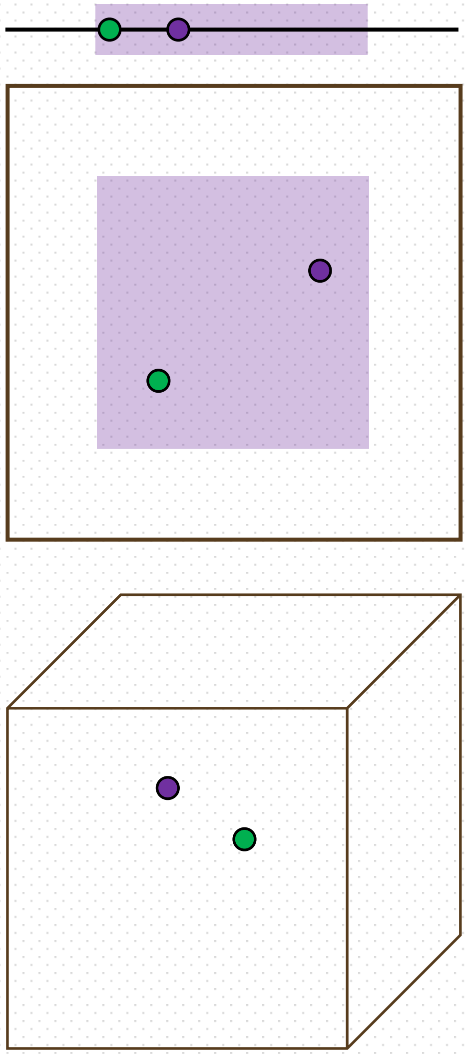
https://en.wikipedia.org/wiki/Bioreactor

- In most real-world systems, the mechanism is to run a high-efficiency risky operation, until safety flags arise, after which, they are redirected to a conservative safe operation

- Second-order or first-order systems
- Safety needs to be ensured (at all steps of optimization) through K Constraints
- Ensuring <u>a single constraint</u> at <u>a single step</u> suffices, in most real-world physical systems

Proposed Environments

Ball-1D, Ball-3D(n-D)

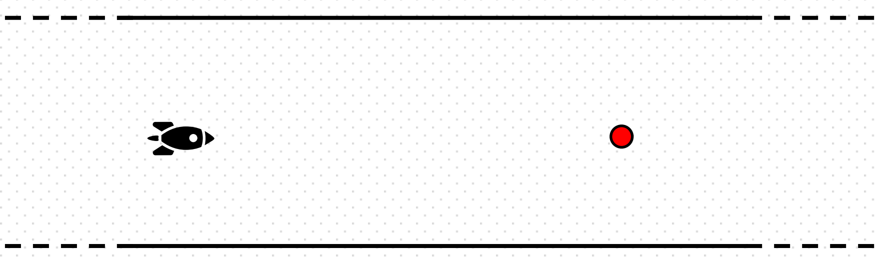


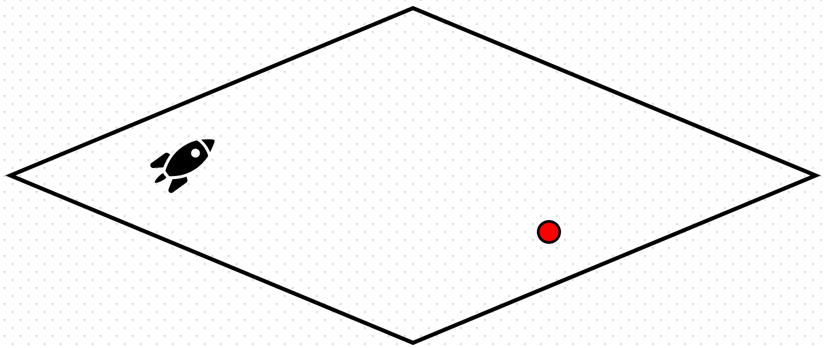
- d-dimensional cube $B_{[a,b]}^{(d)} = \{x | a \le x_i \le b, i = 1, \dots, d\}$
- Goal is to bring the ball(green) to the target(purple), by setting the velocity of the ball at every 4th time step.
- Episode runs for 30s, and the target appears at a uniformly sampled different location every 2s.
- Feasible region for ball, $B_{[0,1]}^{(d)}$, and for target $B_{[0.2,0.8]}^{(d)}$. Episode terminates if ball goes out of bounds.

- State $s = (x_B, v_B, x_T + \epsilon_d)$, where $\epsilon_d \sim \mathcal{N}(0, 0.05 \cdot \boldsymbol{I}_d)$
- Action $a = v_B$
- Dynamics controlled by Newton's laws with damping
- $R(s,a) = \max([1 - 10 \cdot \|x_B - x_T\|_2^2], 0)$
- $\gamma = 0.99$

- First-order system, velocity is controlled to achieve position

Spaceship





- Goal is to bring the spaceship to a fixed target location (red) by controlling thrust engines.
- **First task (Spaceship Corridor):** The safe region is bounded between two infinite parallel walls.
- **Second task (Spaceship Arena):** The safe region is bounded by four walls in a diamond form.
- Termination criteria for episode:
  - ✓ Target is reached.
  - ✓ Spaceship's bow touches the wall.
  - ✓ Time limit (15s for Corridor; 45s for Arena) is exceeded.

- State space: Spaceship's location and velocity
- Action space: $a \in [-1,1]^2$ two thrust engines in front and back, right and left directions.
- Sparse reward structure : 1000 points on reaching the target, 0 everywhere else with $\gamma = 0.99$.

- Second-order system, position controlled by force

CMDP

- The given problem can be modelled as a **CMDP (Constrained Markov Decision Process)**
- Represented as tuple $(S,\ A,\ P,\ R,\ \gamma,\ \mathcal{C})$
- $\mathcal{C} = \{\, c_i : S \times A \to \mathbb{R} \mid i\ \in [K]\}$ represents the **immediate-constraints** functions
- $\bar{\mathcal{C}} = \{\, \overline{c_i} : S \to \mathbb{R} \mid i\ \in [K]\}$ are per-state observation constraints based on the immediate-constraints values.
- $c_i(s,a) \triangleq c(s')$ considering the <u>deterministic</u> nature of policy P s.t. $s' = f(s,a)$
- $\mu : S\ \to A$ represents the deterministic policy

State-wise Constrained Policy Optimization

$$\max_{\theta}\ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t}\, R(s_t, \mu_\theta(s_t))\right]$$

$$\overline{c_i}(s_t) \leq\ C_i\ \forall\ i \in [K]$$

The Solution

Linear Safety-Signal Model

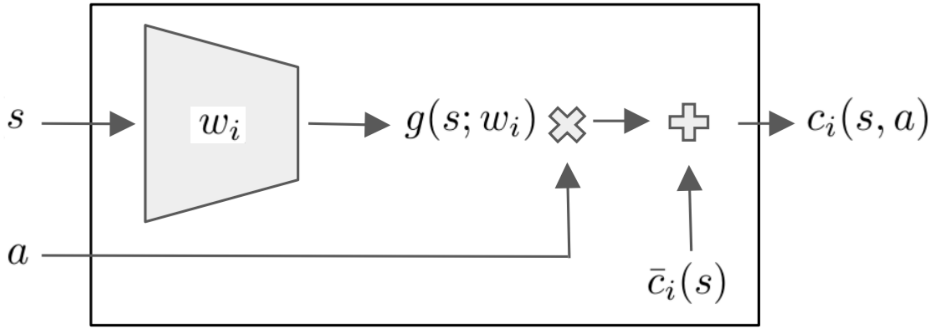
- How does a model learn? Make mistakes(or positive deeds) enough number of times for this negative(positive) effect to propagate in DP methods
- To overcome this, can use single-step dynamics, like transition logs
- The paper suggests to learn just $c_i(s, a)$.
- Can just train a NN that takes in (s, a) and returns c_i

- $c_i(s, a)$ sensitivity varies for different values of *a*, a unified step-size is difficult to obtain
- Solutions to the function have local minima(non-convex), possibility for an incorrect convergence
- K hyperparameters need tuning
- Computationally intensive in-graph computation due to gradient descent after every policy query

$$\underset{a}{\operatorname{argmin}} \left\{ \frac{1}{2} \|a - \mu_\theta(s)\|^2 + \sum_{i=1}^{K} \lambda_i [c_i(s, a) - C_i]^+ \right\}$$

Linear Safety-Signal Model

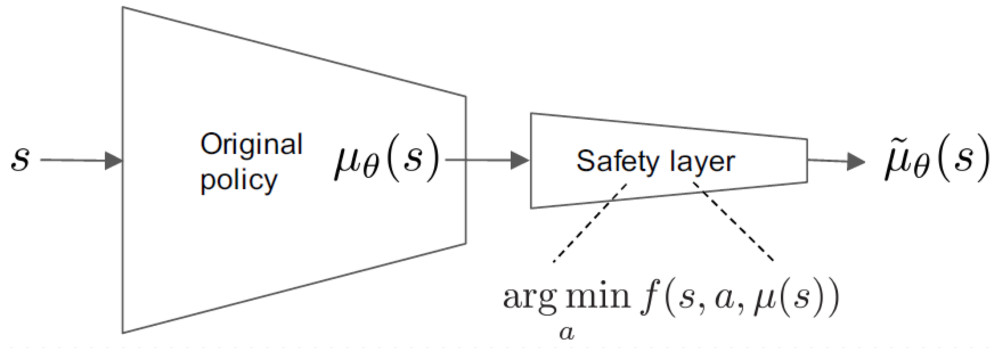
$$\overline{c_i}(s') \triangleq c_i(s,a) \approx \overline{c_i}(s) + g(s;w_i)^T a$$



Training

- Policy-oblivious set $D = \{(s_j, a_j, s'_j)\}$
- $\underset{w_i}{\operatorname{argmin}} \sum_{(s,a,s') \in D} (\overline{c_i}(s') - (\overline{c_i}(s) + g(s;w_i)^T a))^2$
- D can be created by initializing the agent at a random location, and letting it run for some epochs until some time steps, or until a constraint is triggered.
- This training of $g(s;w_i)$ on D is a pre-training task

Safety Layer



- Let $\mu_\theta(s)$ be the deterministic action selected by any deep policy network.
- An additional layer, **the safety layer** is added that aims to solve:

$$\underset{a}{\operatorname{argmin}} \frac{1}{2} \|a - \mu_\theta(s)\|^2 \quad \text{s.t.} \quad c_i(s, a) \leq C_i \ \forall\, i \in [K]$$

- Which reduces to a quadratic objective and linear constraints,

$$\underset{a}{\operatorname{argmin}} \frac{1}{2} \|a - \mu_\theta(s)\|^2 \quad \text{s.t.} \quad \overline{c_i}(s) + g(s; w_i)^T a \leq C_i \ \forall\, i \in [K]$$

- which can be solved by doing $\binom{K}{m}$ checks given K is known to be upper-bounded by m.

- Recall, it is common for only one constraint to be active effectively during a time-step.

- Hence the paper proves the following result.

Safety Layer

Assume there exists a feasible solution to the previous optimization problem, denoted by $\left(a^*, \{\lambda_i^*\}_{i=1}^K\right)$, where λ_i^* is the optimal Lagrange multiplier associated with the i-th constraint. Also assume $|\{i \mid \lambda_i^* > 0\}| \leq 1$, i.e., atmost one constraint is active. Then,

$$\lambda_i^* = \left[\frac{g(s;w_i)^T \mu_\theta(s) + \bar{c}_i(s) - C_i}{g(s;w_i)^T g(s;w_i)}\right]^+$$

and,

$$a^* = \mu_\theta(s) - \lambda_{i^*}^* g(s;w_{i^*})$$

where

$$i^* = \operatorname*{argmax}_i \lambda_i^*$$

Which results in a simple 3 lines of code implementation! A simple projection to a “safe” hyperplane!

Core Components of the Solution Approach

Linear Safety Signal Model

- Develops a linear approximation to model the impact of actions on the system's safety signals.
- Employs neural networks to parameterize this model, learning from historical data how actions influence safety-critical parameters.

Safety Layer

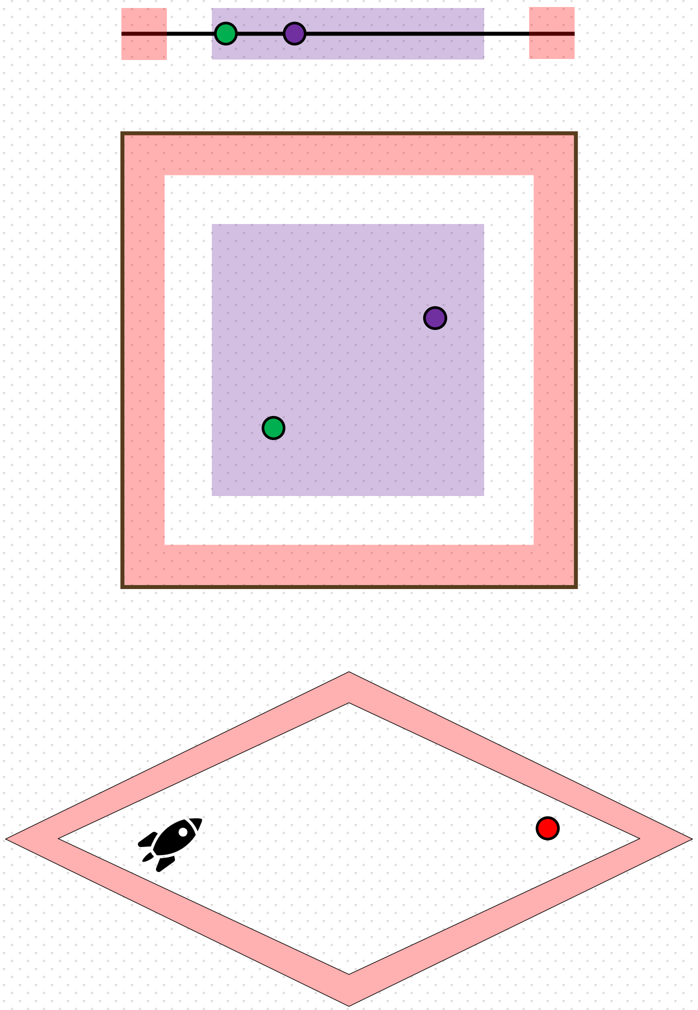
- Integrated with the RL agent's policy network, responsible for evaluating and adjusting proposed actions to ensure safety.
- Ensures that every action taken by the RL agent, after correction, does not lead to constraint violations.

Action Correction Mechanism

- Formulated as a quadratic program with linear constraints, reflecting the linear safety-signal model.
- Can be solved analytically under the assumption that at most one safety constraint is active at a time, providing a closed-form solution for the corrected action.

Experiments and Results

Safety Constraints



- In the previous Ball Environment, some slack is introduced in manoeuvring away from the environment
- $C_i's$ are set to effectively constraint the ball feasible region to $B^{(d)}_{[0.1,0.9]}$ indicated by the red region
- Action correction by linear-safety layer is introduced as ball steps out of this region
- Small gap maintained away from each wall for correction actions before collision happens for Spaceship environment.
- For comparison, gap is set to 0.05 when the distance between the walls in corridor is set to 1

Reward Shaping

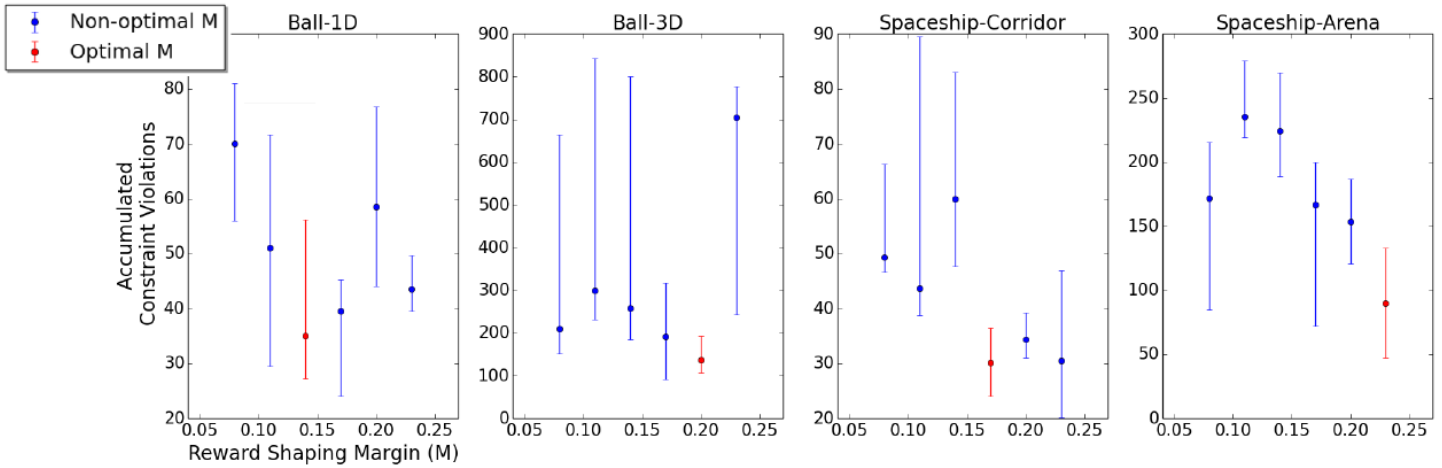
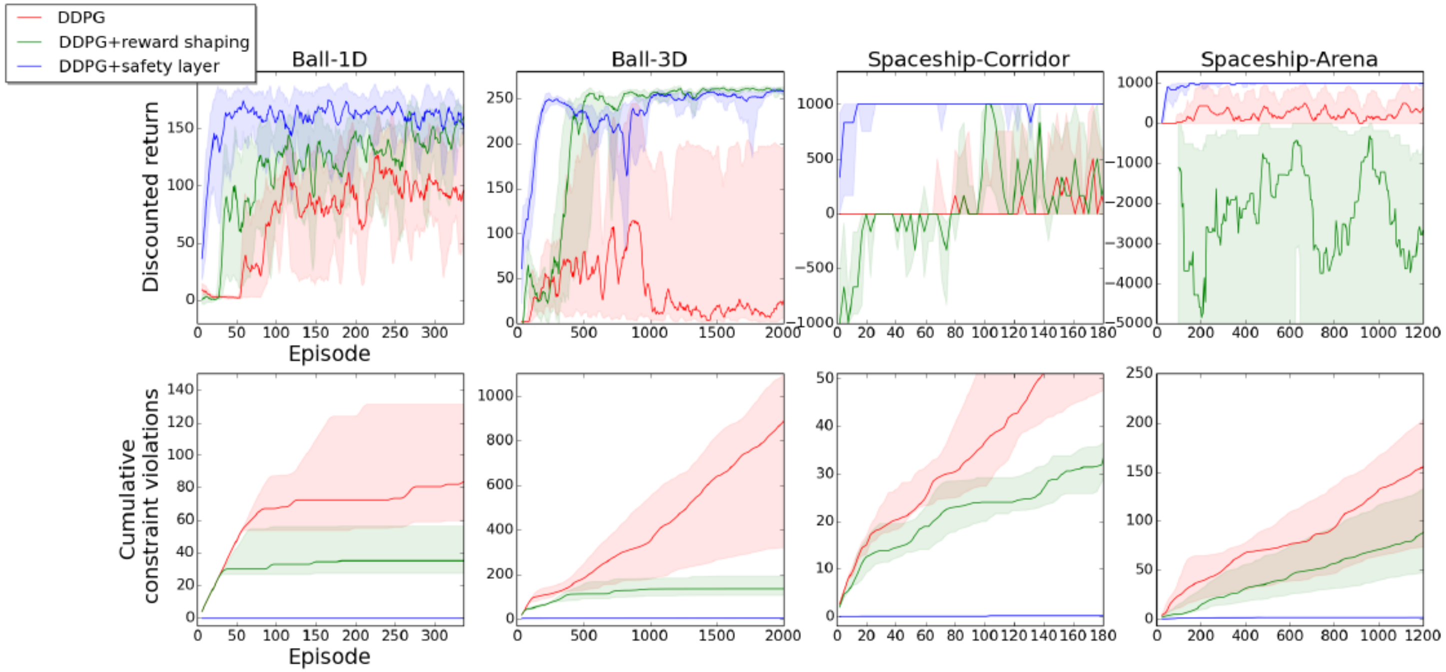


Figure 5. Accumulated constraint violations (lower is better) throughout the training of DDPG+reward shaping, per each task. Plotted are medians with upper and lower quantiles of 10 seeds. The x-axis corresponds to different choice of M – the margin from limits in which reward penalty is incurred. The optimal choice of M is colored red.

- General technique for this problem would be to shape the reward near the boundaries so that the agent avoids them
- Suppose the rewards are negative, with same impact(-1 for BallnD and -1000 for Spaceship), and in a M fraction
- Then, the algorithms are run, with DDPG as the base model, varying M ∈ {0:08; 0:11; 0:14; 0:17; 0:2; 0:23}
- Noticeably, there are always constraint violations

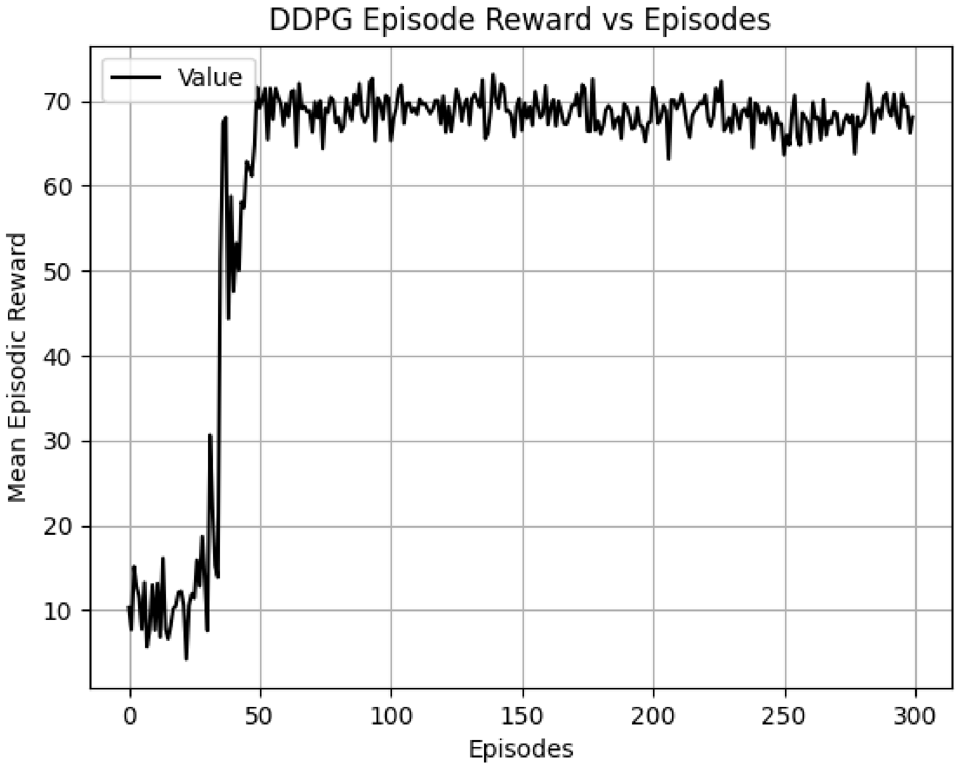
Reward Shaping

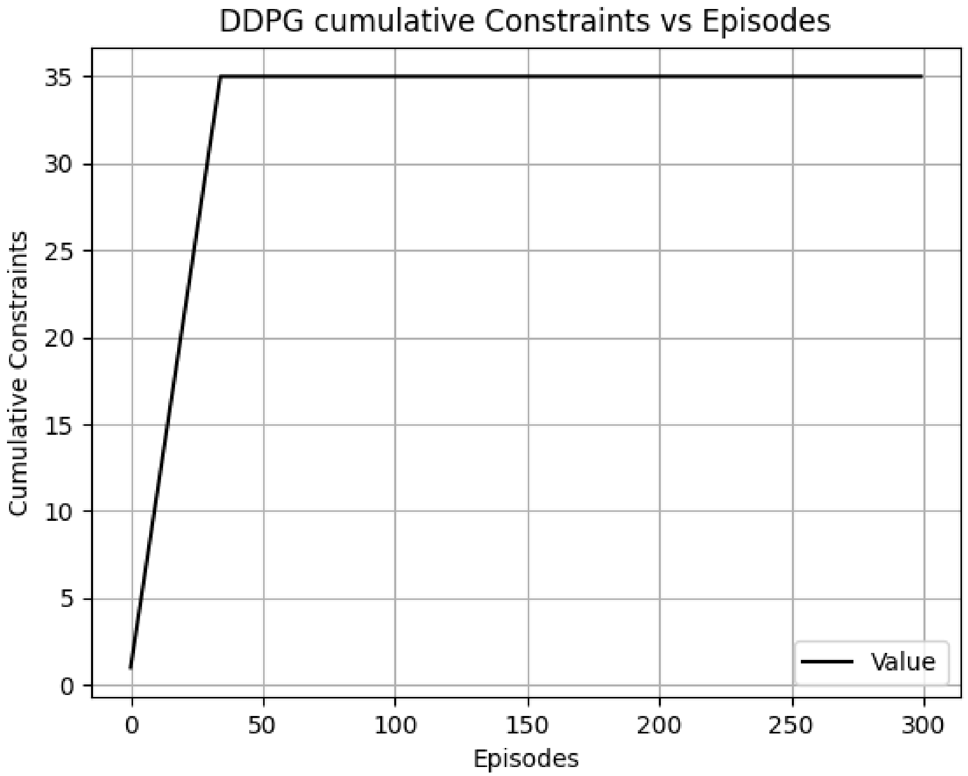


- DDPG + Reward Shaping performs better than DDPG in Ball-nD environments.
- DDPG + safety layer performs way better, with 0 constraint violations
- DDPG doesn't converge to any reasonable policy at all in Spaceship. Reward-shaping had a negative impact.
- DDPG + safety layer still proved the best, with a convergence which is very early, and 0-1 constraint violations.

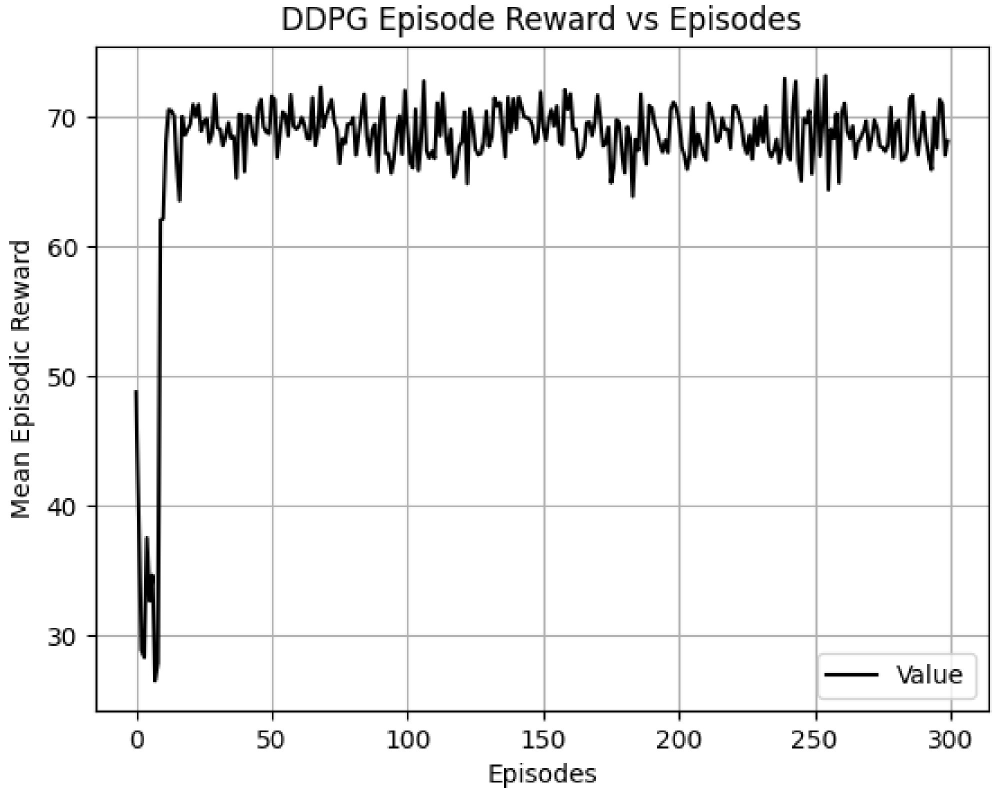
Our Implementation

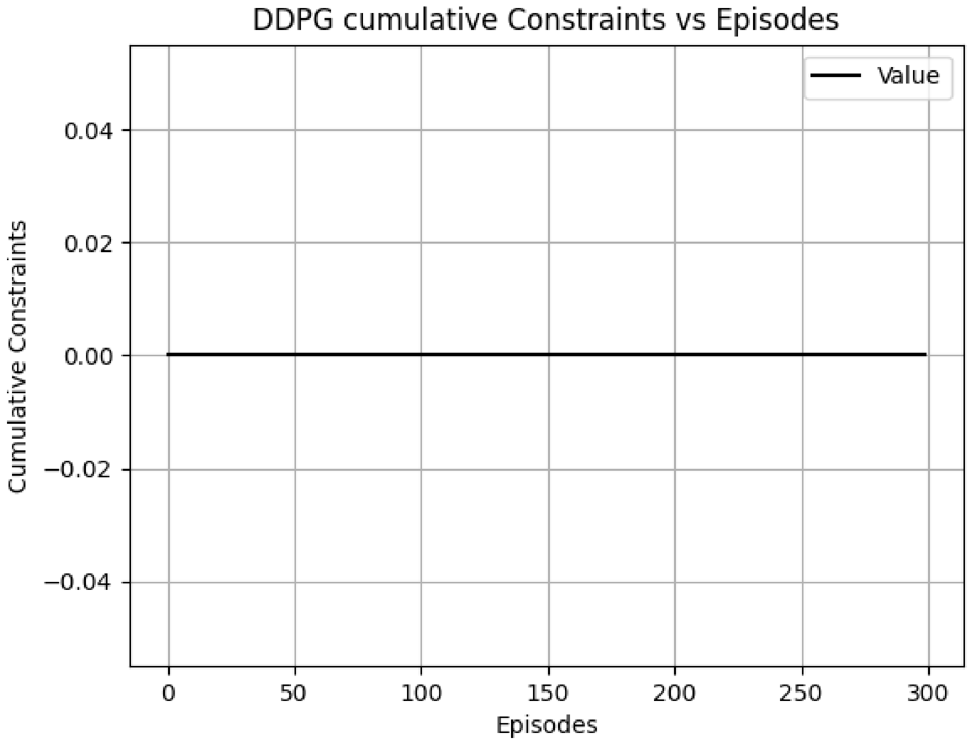
Results on Ball(n-D) (DDPG layer only)



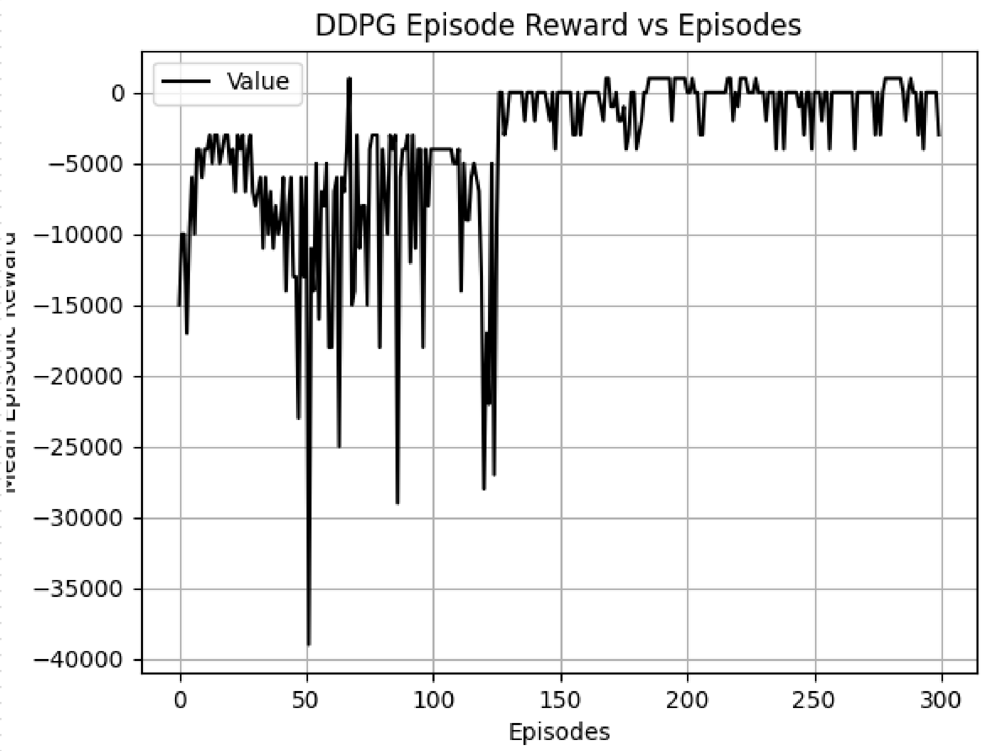


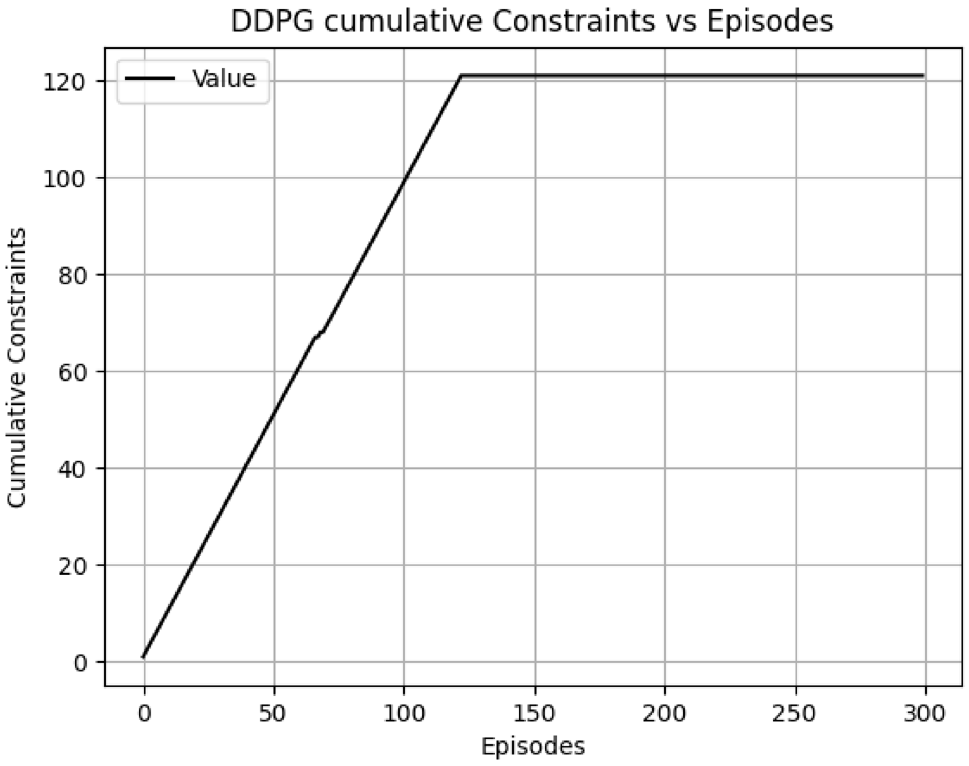
Results on Ball(n-D) (DDPG + safety layer only)



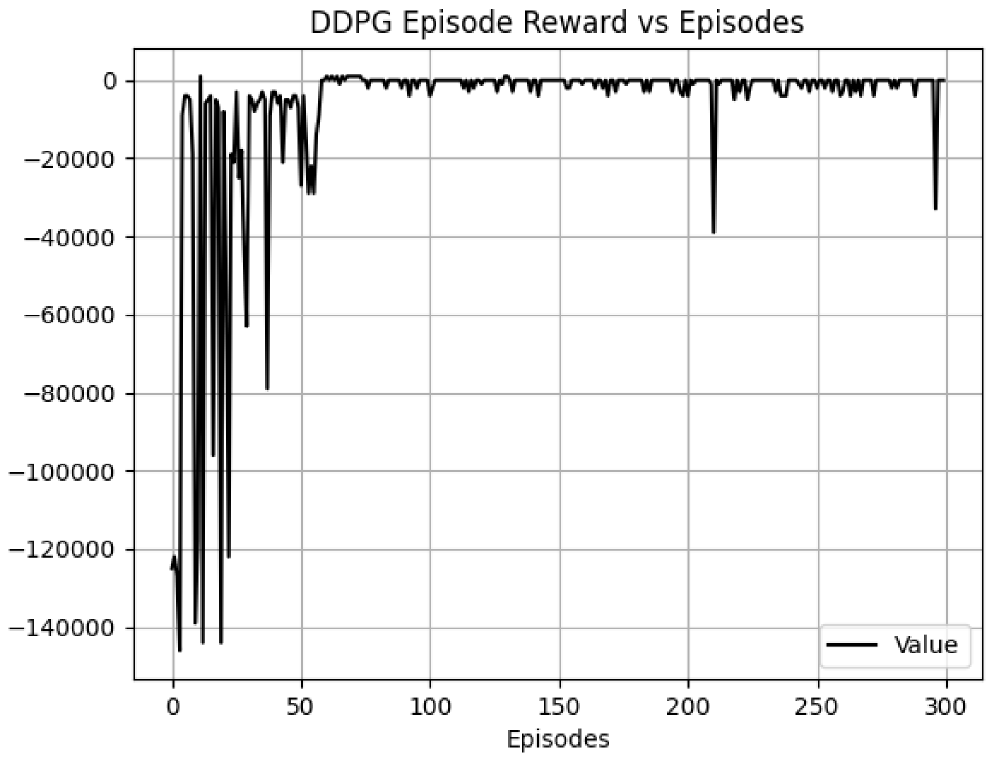


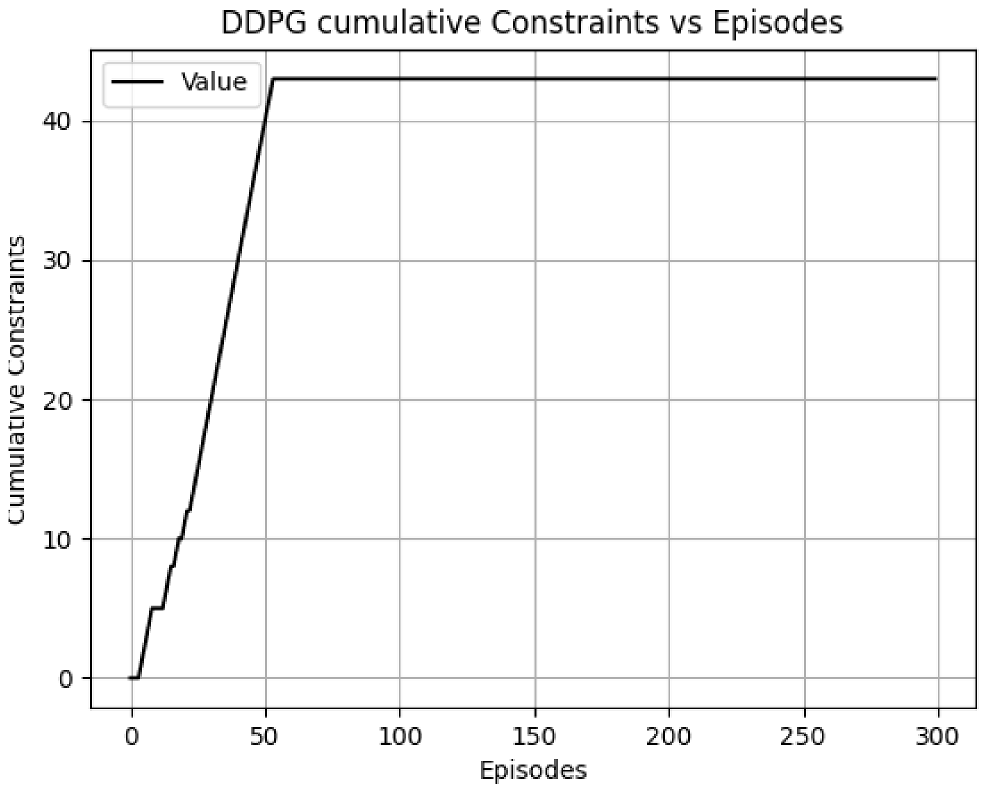
Results on Spaceship Corridor (DDPG only)





Results on Spaceship Corridor (DDPG + safety layer only)





Open-source implementations can be found at:

1. https://github.com/kollerlukas/safe-explorer
2. https://github.com/AgrawalAmey/safe-explorer

Proposed Work

- As far as these environments go, the solution is almost perfect(zero constraint violations).
- So, the first motive is to look, and develop environments, where these ideas fail.
  - Either through the NN not being able to learn the $g(s; w_i)$ function well
  - Or, through multiple constraints being broken at the same time.
- The paper also mentions the latter as a possible pain-point, with scope for further improvement.
- Another possible work could be to try and actively try this problem in real and physical world environments(say data center cooling, or robotic actuation).

Thanks for Listening!

P.S. The paper provides some nice visualizations of the agent learning in these environments.
https://youtu.be/KgMvxVST-9U
https://youtu.be/yr6y4Mb1ktI