

A APPENDIX

A.1 SAFE POLICY UPDATE UNDER THE LYAPUNOV CONSTRAINT

Let the safe initial (baseline) policy be given by π_B and the Lyapunov function be defined as follows:

$$\mathcal{L}_{\pi_B}(s_0, d_0) = \{L : S \rightarrow R_{\geq 0} : \mathcal{B}_{\pi_B, c}[L](s) \leq L(s), \forall s \in S; L(s_0) \leq d_0\} \quad (16)$$

where c is the immediate cost function. Lyapunov functions depends on the safe baseline policy, an initial state and the cost constraint, and have the property that a one-step Bellman operator produces a value that is less than the value of the function at each state. We also know that the cost value function belongs to the set and hence the set is non-empty. Consider any Lyapunov function $L_{\pi_B} \in \mathcal{L}_{\pi_B}(s_0, d_0)$ and define:

$$\mathcal{I}_{L_{\pi_B}} = \{\pi(\cdot|s) \in \mathcal{P} : \mathcal{B}_{\pi, c}[L_{\pi_B}](s) \leq L_{\pi_B}(s) \forall s\} \quad (17)$$

to be set of policies consistent with the Lyapunov function L_{π_B} , called L_{π_B} -induced policies. These are the set of policies ($\pi \in \mathcal{I}_{L_{\pi_B}}$) for which a Bellman Operator $\mathcal{B}_{\pi, c}$ on a state s produces a value that is less than the value of function at that state $L_{\pi_B}(s)$

Note that $\mathcal{B}_{\pi, c}$ is a contraction mapping, so we have

$$V_{\pi}^c(s) = \lim_{k \rightarrow \infty} \mathcal{B}_{\pi, c}^k[L_{\pi_B}](s) \leq L_{\pi_B}(s) \forall s \in \mathcal{S} \quad (18)$$

From the definition of Lyapunov function, we also have that $L_{\pi_B}(s_0) \leq d_0$. This implies that any policy induced by the Lyapunov function, i.e. policies in the L_{π_B} -induced policy set, are “safe” i.e. $V_{\pi}^c(s_0) = D_{\pi}(s_0) < d_0$. The method for safe reinforcement learning then searches for the highest performing policy within the safe policies defined by the set of L_{π_B} -induced policies. The objective here then is to design a Lyapunov function which contains the optimal policy, i.e optimal policy belongs to the set of L_{π_B} -induced policies so that the optimization restricted in this set indeed results in the solution of Eq. 2.

In general, the optimal policy π^* does not belong the policies induced by the Lyapunov functions. Chow et al. (2018) show that without loss of optimality, the Lyapunov function that contains the optimal policy in its L_{π_B} -induced policy set can be expressed as $L_{\pi_B, \epsilon}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (c(s_t) + \epsilon(s_t)) | \pi_B, s]$, where $\epsilon(s_t) \geq 0$. The function $L_{\pi_B, \epsilon}(s)$ can be thought of as a cost-value function for policy π_B augmented by an additional per-step cost $\epsilon(s_t)$. First, it can be verified that π_B is indeed, in the set of L_{ϵ} -induced policies:

$$L_{\pi_B, \epsilon}(x) = \mathcal{B}_{\pi_B, c+\epsilon}[L_{\pi_B, \epsilon}](s) \geq \mathcal{B}_{\pi_B, c}[L_{\pi_B, \epsilon}](s) \quad (\epsilon(s_t) > 0 \forall s_t). \quad (19)$$

It was shown in Chow et al. (2018) that finding a state dependent function ϵ such that the the optimal policy is inside the corresponding $L_{\pi_B, \epsilon}$ -induced set is generally not possible and requires knowing the optimal policy. As an approximation, they suggest to create the Lyapunov function with the largest auxiliary cost $\hat{\epsilon}$, such that $L_{\pi_B, \hat{\epsilon}}(s) \geq \mathcal{B}_{\pi_B, c}[L_{\pi_B, \hat{\epsilon}}](s)$ and $L_{\pi_B, \hat{\epsilon}}(s_0) \leq d_0$. The first condition is satisfied as shown in Eq. 19 when $\hat{\epsilon}(s) \geq 0 \forall s$ and the second condition can be satisfied by the following derivation. Bold letters are used to denote vectors and $\mathbf{P}_{s, s'}^{\pi_B}$ is the transition probability matrix from state s to s' under policy π_B . The vectors contain the function value at each state.

$$L_{\pi_B, \hat{\epsilon}} = \mathbf{d} + \hat{\epsilon} + \gamma \mathbf{P}_{s, s'}^{\pi_B} L_{\pi_B, \hat{\epsilon}} \quad (20)$$

$$L_{\pi_B, \hat{\epsilon}} = (\mathbf{I} - \gamma \mathbf{P}_{s, s'}^{\pi_B})^{-1} \mathbf{d} + (\mathbf{I} - \gamma \mathbf{P}_{s, s'}^{\pi_B})^{-1} \hat{\epsilon} \quad (21)$$

$$\mathbf{1}(s)^T L_{\pi_B, \hat{\epsilon}} = \mathbf{1}(s)^T D_{\pi_B} + \mathbf{1}(s)^T (\mathbf{I} - \gamma \mathbf{P}_{s, s'}^{\pi_B})^{-1} \hat{\epsilon} \quad (22)$$

where $\mathbf{1}(s)^T$ is a one-hot vector in which the non-zero unit element is present at s . To ensure that the cumulative cost at the starting state is less than the constraint threshold, using Eq 22 we have:

$$\begin{aligned} L_{\pi_B, \hat{\epsilon}}(s_0) &\leq d_0 \\ D_{\pi_B}(s_0) + \mathbf{1}(s_0)^T (\mathbf{I} - \gamma \mathbf{P}_{s, s'}^{\pi_B})^{-1} \hat{\epsilon} &\leq d_0 \end{aligned}$$

Notice that $\mathbf{1}(s_0)^T(I - \gamma \mathbf{P}_{s,s'}^{\pi_B})^{-1} \mathbf{1}(s)$ represents the total discounted visiting probability $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}(s_t = s) | s_0, \pi_B]$ of any state s from the initial state s_0 . Restricting $\hat{\epsilon}$ to be a constant function w.r.t state for simplicity, the value of $\hat{\epsilon}$ can be upper-bounded as:

$$\hat{\epsilon}(s) \leq (d_0 - D_{\pi_B}(s_0)) / \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \right] = (1 - \gamma)(d_0 - D_{\pi_B}(s_0)) \quad (23)$$

In summary, a Lyapunov function is obtained such that optimizing policies in the $L_{\pi_B, \hat{\epsilon}}$ -induced set of policies, safety is ensured. For any policy π to lie in the $L_{\pi_B, \hat{\epsilon}}$ -induced set the following condition needs to hold $\forall s \in \mathcal{S}$:

$$L_{\pi_B, \hat{\epsilon}}(s) \geq T_{\pi, d}[L_{\pi_B, \hat{\epsilon}}(s)] \\ d(s) + \hat{\epsilon}(s) + \gamma \sum_a \pi_B(a|s) \left(\sum_{s'} P(s'|s, a) L_{\pi_B, \hat{\epsilon}}(s') \right) \geq d(s) + \gamma \sum_a \pi(a|s) \left(\sum_{s'} P(s'|s, a) L_{\pi_B, \hat{\epsilon}}(s') \right)$$

We can simplify further to get:

$$\begin{aligned} \hat{\epsilon}(s) &\geq \left(\sum_a (\pi(a|s) - \pi_B(a|s)) \right) \left[\gamma \sum_{s'} P(s'|s, a) L_{\pi_B, \hat{\epsilon}}(s') \right] \\ \hat{\epsilon}(s) &\geq \left(\sum_a (\pi(a|s) - \pi_B(a|s)) \right) \left[\gamma \sum_{s'} P(s'|s, a) L_{\pi_B, \hat{\epsilon}}(s') + d(s) + \hat{\epsilon}(s) \right] \\ \hat{\epsilon}(s) &\geq \left[\sum_a (\pi(a|s) - \pi_B(a|s)) Q_{L_{\pi_B, \hat{\epsilon}}}(s, a) \right] \end{aligned}$$

where

$$Q_{L_{\pi_B, \hat{\epsilon}}}(s, a) = d(s) + \hat{\epsilon}(s) + \gamma \sum_{s'} P(s'|s, a) L_{\pi_B, \hat{\epsilon}}^{\pi}(s') \quad (24)$$

This can be extended to continuous action spaces to get the following objective:

$$\pi_+(\cdot|s) = \max_{\pi \in \mathcal{P}} J_{\pi}(s_0), \text{ s.t. } \int_{a \in \mathcal{A}} (\pi(a|s) - \pi_B(a|s)) Q_{L_{\pi_B, \hat{\epsilon}}}(s, a) \leq \hat{\epsilon}(s) \quad (25)$$

Using the Lyapunov function, the trajectory-based constraints of CMDP are converted to a per-state constraint (Eq.25), which are often much easier to deal with.

In the case of deterministic policy, the policy update becomes:

$$\pi_+(\cdot|s) = \max_{\pi \in \mathcal{P}} J_{\pi}(s_0), \text{ s.t. } Q_{L_{\pi_B, \hat{\epsilon}}}(s, \pi(s)) - Q_{L_{\pi_B, \hat{\epsilon}}}(s, \pi_B(s)) \leq \hat{\epsilon}(s) \quad (26)$$

An intuitive way to understand the constraint in deterministic policies is to see that at every timestep we are willing to tolerate an additional constant cost of ϵ compared to the baseline safe policy. At the start state, the maximum increase in expected cost will be $\sum_{t=0}^{\infty} \gamma^t \epsilon = \frac{\epsilon}{1-\gamma}$. We want that the new expected cost be less than the threshold, i.e. $D_{\pi}(s_0) + \frac{\epsilon}{1-\gamma} \leq d_0$ which gives us the Lyapunov constraint equation.

A.1.1 FROM LYAPUNOV FUNCTIONS TO COST Q FUNCTIONS

Using the definition of $Q_{L_{\pi_B, \hat{\epsilon}}}(s, a)$ from Eq. 5 and when $\hat{\epsilon}(s)$ is a constant function (denote by $\hat{\epsilon}$), we can replace $Q_{L_{\pi_B, \hat{\epsilon}}}$ by $Q_{\pi_B}^C$,

$$\begin{aligned} Q_{L_{\pi_B, \hat{\epsilon}}}(s, a) &= c(s) + \hat{\epsilon} + \gamma \sum_{s'} P(s'|s, a) L_{\pi_B, \hat{\epsilon}}(s') \\ &= c(s) + \hat{\epsilon} + \left[\gamma \sum_{s'} P(s'|s, a) [c(s') + \hat{\epsilon} + \sum_{s''} P_{(s''|s')}^{\pi_B}(L_{\pi_B, \hat{\epsilon}}(s''))] \right] \\ &= \sum_{t=0}^{\infty} \gamma^t \hat{\epsilon} + \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t c(s_t) | \pi_B, a_0 = a, s_0 = s \right] \\ &= \sum_{t=0}^{\infty} \gamma^t \hat{\epsilon} + Q_{\pi_B}^C(s, a) \end{aligned}$$

which is the cost Q function, since the Lyapunov function $Q_{L_{\pi_B, \hat{\epsilon}}}(s, a)$ and the cost-Q function $Q_{\pi_B}^C(s, a)$ only differ by a constant ($\sum_{t=0}^{\infty} \gamma^t \hat{\epsilon}$).

A.2 ADDITIONAL RESULTS

A.2.1 BENCHMARKS ON OPENAI SAFETY GYM

In this section, we present the training curves for all the OpenAI safety gym environments with Point and Car robot. Figure A.2.1 shows the Average Cost and Average return for these environments. The dotted red line indicates the constraint threshold which is kept to be 25 across all environments. We observe that LBPO rarely violates constraint during training. Table 3 shows the raw cumulative returns of the converged policy for different methods on the safety environments. We average all results over 3 random seeds.

We observe that in tasks with Doggo robot, none of the methods are able to obtain good performing policy. We attribute this to be the difficulty of Doggo environments, involving an inherent tradeoff of reward with cost. In the environment PointGoal2, we are unable to obtain safe policies even when training an RL agent solely on the cost objective. LBPO still outperforms baselines for constraint satisfaction on this environment.

Method	PPO	PPO-lagrangian	CPO	SDDPG	BACKTRACK	LBPO
PointGoal1	22.99	19.00	10.26	10.45	15.54	11.06
PointGoal2	23.04	4.60	-0.37	-0.08	1.04	0.61
PointPush1	4.61	3.04	1.73	2.71	2.43	3.15
PointPush2	2.15	1.04	0.46	0.48	0.89	0.77
CarGoal1	34.62	15.55	2.76	3.38	17.22	13.03
CarGoal2	26.70	1.78	4.60	5.74	4.35	5.62
CarPush1	3.89	2.72	-3.13	1.69	3.38	1.89
CarPush2	2.03	0.72	0.82	0.75	0.81	0.94
DoggoGoal1	38.76	-0.65	0.14	0.10	0.15	0.28
DoggoGoal2	18.38	0.31	0.04	0.06	0.06	0.06
DoggoPush1	0.82	0.07	0.01	0.00	0.06	0.01
DoggoPush2	1.10	0.08	-0.00	-0.00	0.07	-0.01

Table 3: Cumulative unnormalized return of the converged policy for each safety algorithm. LBPO tradeoffs return for better constraint satisfaction. Bold numbers show the best performance obtained by a safety algorithm (thus excluding PPO).

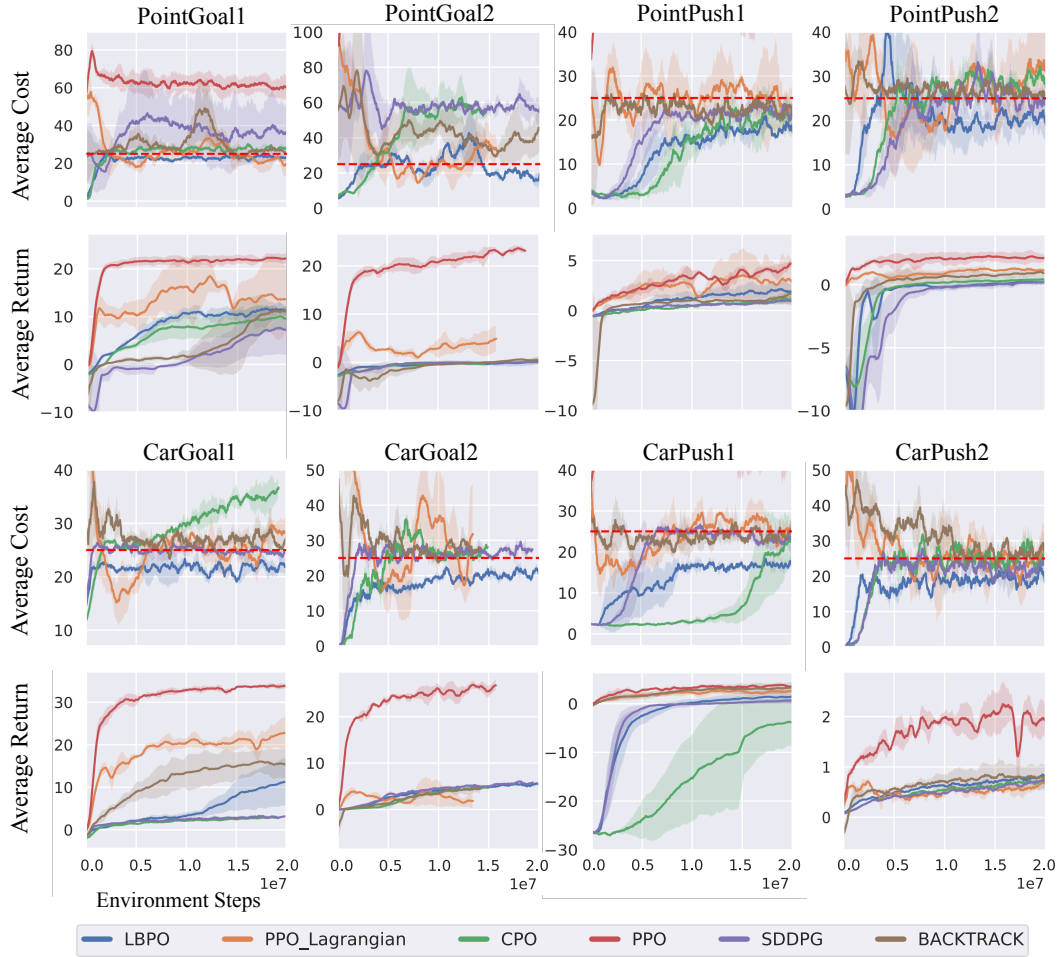


Figure 6: Training curved for LBPO in comparison to baselines: PPO, PPO-lagrangian, CPO, SDDPG. We also compare against our simple baseline BACKTRACK here. For each environment, the top row shows the Average undiscounted cumulative cost during training, and bottom row shows the Average undiscounted return. PPO often has large constraint violations and is clipped from some plots, when its constraint violations are high. Red dashed line in Average Cost plots shows the constraint limit which is 25 in all environments.

A.2.2 BACKTRACKS IN CPO AND SDDPG

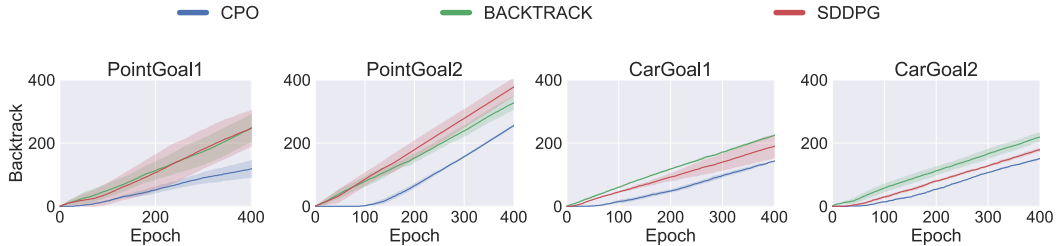


Figure 7: We compare the cumulative number of backtracking steps taken by CPO and SDDPG to BACKTRACK method for the first 400 epochs/policy updates.

Figure 7 shows the cumulative number of backtracks performed by each method CPO, SDDPG, BACKTRACK during the first 400 policy update steps. We see the CPO and SDDPG performs a high number of backtracks, often comparable to the method BACKTRACK which relies explicitly on backtracking for safety.

A.3 IMPLEMENTATION DETAILS

In LBPO, Q-functions (both reward and cost) have network architecture comprising of two hidden layers of 64-hidden size each. The policy is also a multilayer neural network comprising of three hidden layers of 256 units each. LBPO policies are deterministic and have a fixed exploration noise in the action space given by $\mathcal{N}(0, 0.05)$. Our trust region update for the policy takes into account the exploration noise which makes our behavior deployment policy stochastic. We use $N = 30$ trajectories each of 1000 horizon length for generating our on-policy samples. These samples are used for estimation of $\hat{\epsilon}$ and evaluating the Q functions. We update the policy under a trust region followed by a line search with exponential decay which ensures that the resulting update is indeed satisfying the KL constraint as well the safety Lyapunov constraint. We do a hyperparameter search for β in the set $[0.005, 0.008, 0.01, 0.02]$ to find the best tradeoff between cost and reward and observe that a value of 0.005 works well across most environments. PointGoal1, PointPush1, PointPush2, CarGoal1, CarPush1, CarPush2, DoggoGoal1, DoggoPush1, DoggoPush2 use beta value of 0.005. CarGoal2 and DoggoGoal2 uses value of 0.008 and Pointgoal2 uses beta value of 0.01. We ignore the barrier loss if β is sufficiently low. We call this parameter β -thres and it is set to 0.05 across all environments.

Algorithm 1: LBPO

- 1 Initialize parameterized actor π_ϕ with a safe initial policy, reward Q-function Q_θ^R and a cost Q function Q_θ^C
 - 2 **for** $i \leftarrow 1$ **to** $Iter$ **do**
 - 3 **Step 1:** Collect N trajectories $\{\tau\}_{j=1}^N$ using the safe policy $\pi_{\phi, i-1}$ from previous iteration $i - 1$.
 - 4 **Step 2:** Using the on-policy trajectories, evaluate the reward Q-function and the cost-function, by minimizing the respective bellman residual of the TD- (λ) estimate.
 - 5 **Step 3:** Update the policy parameters by minimizing the objective in Eq 12.

$$\min_{\phi} \mathbb{E}_{s \sim \mathcal{R}} \left[-Q_{\pi_{\phi, i-1}}^R(s, \pi_{\phi}(s)) + \psi(Q_{\pi_{\phi, i-1}}^C(s, \pi_{\phi}(s))) \right]$$

$$\text{s.t } D_{\text{KL}}(\pi_{\phi} + \mathcal{N}(0, \delta) \| \pi_{\phi, i-1} + \mathcal{N}(0, \delta)) < \mu$$
 - 6 **Step 4:** Set $\pi_{\phi, i}$ to be the safe policy resulting from the update in Step 3 π_{ϕ} .
 - 6 **end**
-

We obtain safe initial policies for benchmarking by pretraining the policy using standard RL methods to minimize the cumulative cost. Although this strategy is usually not suitable for deployment in real-world as the pretraining might itself violate safety constraints, we can use simple hand-designed safe policy for initializing the method in real-world experiments.

To ensure fair comparison across methods, we use the same safe initial policy for each of the safety methods. Note that, our results for CPO (Achiam et al., 2017) significantly differ from the benchmarks shown in (Ray et al., 2019) due to the fact that we initialize CPO from safe policy contrary to their approach. We also keep the same policy architecture across methods although CPO, PPO and PPO-lagrangian uses policies with learned variance so as to replicate the original behavior of these methods.

Table 4: LBPO Hyperparameters

Hyperparameter	Value
N	30
β	0.005 ¹
β -thres	0.05
Policy learning rate	3e-4
Q-function learning rate	1e-3
Trust region (μ)	0.012
λ	0.97
δ	0.05
Horizon	1000

We implement our version of SDDPG which uses the α -projection technique as shown in (Chow et al., 2019). A brief discussion of practical issue faced in the implementation is present in Section 4. We use behavior cloning to distill the policy with the projection layer into a parameterized multilayer perceptron policy. We run 100 iterations of behavior cloning with learning rate of 0.001. We implement a line search with exponential decay in parameter space to ensure that the resulting update do not violate the Lyapunov constraints to incorporate additional safety. We use similar policy architecture as LBPO for α -SDDPG.

¹ β is set to 0.005 for most environments. Appendix A.3 describes specific value of β for each environment.