Zero-Shot Constraint Satisfaction with Forward-Backward Representations

Anonymous authors
Paper under double-blind review

Abstract

Traditionally, constrained policy optimization with Reinforcement Learning (RL) requires learning a new policy from scratch for any new environment, goal or cost function, with limited generalization to new tasks and constraints. Given the sample inefficiency of many common deep RL methods, this procedure can be impractical for many real-world scenarios, particularly when constraints or tasks are changing. As an alternative, in the unconstrained setting, various works have sought to pre-train representations from offline datasets to accelerate policy optimization upon specification of a reward. Such methods can permit faster adaptation to new tasks in a given environment, dramatically improving sample efficiency. Recently, zero-shot policy optimization has been explored by leveraging a particular *forward-backward* decomposition of the successor measure to learn compact, task-agnostic representations of the environment dynamics. However, these methods have been primarily studied in the unconstrained setting. In this work, we introduce a method for performing zero-shot *constrained* policy optimization from forward-backward representations. We introduce a principled inference-time procedure for zero-shot constrained policy optimization and demonstrate its empirical performance on illustrative environments. Finally, we show that even in simple environments, there remains an optimality gap in zero-shot constrained policy optimization, inviting future developments in this area.

1 Introduction

Real-world autonomous systems must typically obey safety constraints, motivating a large body of work in constrained reinforcement learning (RL). However, existing methods for constrained RL have typically only been explored in the context of tabula rasa RL, i.e., learning an RL policy from scratch. These methods operate either in an online environment with cost feedback (Achiam et al., 2017; Zhang et al., 2020; Xu et al., 2021), or on an offline dataset with cost annotations (Lee et al., 2022; Xu et al., 2022; Liu et al., 2023). In such cases, the learned policy is valid *only* under the particular reward and cost function under which it was trained. However, comprehensive descriptions of a system's rewards and constraints may not be available during training, and users may also introduce new tasks or constraints after training is complete. Ideally, our methods would support the zero-shot generation of policies for these new tasks and constraints without retraining.

In this regard, much effort has gone towards developing methods for pretraining representations in RL, either online (Pathak et al., 2017; Liu & Abbeel, 2021) or offline (Stooke et al., 2021; Schwarzer et al., 2021), to instantiate faster learning once the task is specified. So-called *forward-backward* (FB) representations (Touati & Ollivier, 2021) are one such paradigm for representation learning that offers promise for zero-shot policy generalization across tasks. FB representations consist of a linear *reward function encoder*, which maps reward functions to finite-dimensional embeddings, and an embedding-conditioned policy that optimizes the embedded reward. For any unconstrained RL problem, one can thus leverage a pretrained FB representation to perform zero-shot policy optimization by first using it to embed the relevant reward function, before deploying the policy that results from conditioning on this embedding. While these methods have found considerable success in accelerating RL in the unconstrained setting , relatively little attention has been given to how these representations perform on constrained tasks.

In principle, these same representations can be directly applied to perform constrained policy optimization by supplying reward functions that penalize cost-incurring states. However, these naive adaptations do not necessarily yield satisfying solutions. Namely, the effect of scaling the cost penalty to ensure constraint satisfaction is not necessarily benign—as it increases, the resulting embeddings become *out of distribution* relative to the sampled embeddings that are seen over the course of FB pretraining, and consequently, the resulting embedding-conditioned policies are unreliable. Instead, as we explore in this work, by leveraging the linearity of the reward encoder in the FB representation, an appropriate cost multiplier can be determined by a Lagrangian approach. By determining a multiplier for the cost embedding in this way, one can, in theory, assert that the policy conditioned on the difference between the reward embedding and the scaled cost embedding will be an optimal policy for the constrained MDP.

The primary contributions of our work are as follows:

1. We introduce a principled inference-time procedure for performing zero-shot *constrained* policy optimization, which does not require any online environment interactions, by extending zero-shot policy optimization with forward-backward representations.
2. Empirically, we demonstrate that our proposed approach can select an appropriate cost multiplier which prevents most constraint violations without severely impacting task rewards.
3. Finally, we demonstrate that there still remains a gap to the performance of a post-hoc procedure which uses privileged information from online evaluations to determine the cost multiplier, inviting further research in this important area.

2 Related Work

Constrained RL seeks to learn policies that are optimal under a reward function while respecting constraints on one or more cost functions. Many methods for online constrained RL have been proposed (Achiam et al., 2017; Zhang et al., 2020; Xu et al., 2021), though Lagrangian adaptations of popular RL algorithms remain common (Stooke et al., 2020). Constrained RL has also been explored in the offline setting, where the dataset is assumed to be annotated with both costs and rewards (Lee et al., 2022; Xu et al., 2022; Liu et al., 2023). Relatively little work has explored zero-shot or few-shot constrained RL. Yao et al. (2023) train a policy which can be adapted zero-shot at inference time to different cost thresholds, however, this is limited only to varying thresholds and not wholly different cost functions. Touati & Ollivier (2021) consider some simple constraints and show zero-shot performance is possible given an appropriate cost multiplier, however they do not offer a method for determining this multiplier offline for constraint satisfaction.

Zero-shot reinforcement learning has been explored extensively in the unconstrained case. In the tabular case, successor representations can be computed, permitting zero-shot value estimation for a fixed policy (Dayan, 1993). This representation generalizes to the continuous MDP case via the successor *measure* (Blier et al., 2021), which has been used for zero-shot value estimation (Janner et al., 2020) and return distribution estimation (Wiltzer et al., 2024) via generative modelling. In the continuous case, one body of work seeks to perform unsupervised (reward free) pre-training to learn generally useful representations (Eysenbach et al., 2021). Such methods can be combined with successor feature (SF) methods (Barreto et al., 2017), a generalization of the successor representation, to perform zero-shot RL (Borsa et al., 2018; Touati et al., 2022), however, such methods require separately training a feature representation which can hinder zero-shot performance. Notably, Touati & Ollivier (2021) introduces the *forward-backward* representation as a particular decomposition of the successor measure into one component that acts as a reward function encoder, and another which induces a reward-embedding-conditioned policy that is greedy with respect to its action-values. Particularly, their approach implicitly learns features through a successor measure consistency loss, jointly with task-conditioned optimal policies. Touati et al. (2022) show that this method outperforms explicit feature learning methods at zero-shot policy optimization. However, to our knowledge, no work has explicitly considered these representations in the constrained RL setting.

3 Background

Before delving into our approach for imbuing zero-shot policy optimization models with the ability to satisfy constraints, we outline the formalisms of the (constrained) RL setting and the forward-backward representation, which serve as the backbone of our work.

In the sequel, we operate in a Markov Decision Process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P : \mathcal{S} \times \mathcal{A} \to \mathscr{P}(\mathcal{S})$, a bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution $\rho_0 \in \mathscr{P}(\mathcal{S})$, and discount factor $\gamma \in [0, 1)$. A *policy* is a mapping $\pi : \mathcal{S} \to \mathscr{P}(\mathcal{A})$ which maps state observations to (possibly random) actions. Nominally, the objective of RL is to find a policy $\pi^\star$ that maximizes the expected discounted return,

$$\pi^\star \in \operatorname*{argmax}_\pi \mathbb{E} \sum_{t\geq 0} \gamma^t r(S_t, A_t), \quad S_{t+1} \sim P(\cdot \mid S_t, A_t),\ A_t \sim \pi(\cdot \mid S_t), S_0 \sim \rho_0. \tag{1}$$

For reasons that will be apparent in the next section, we will refer to the objective (1) as the *unconstrained* RL problem.

3.1 Constrained RL

In constrained RL, along with a reward function r, the agent is presented with an instantaneous bounded cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and must optimize

$$\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \text{ subject to } \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right] \leq \beta, \tag{2}$$

where β is a given budget. The Lagrangian dual of Equation 2 is given by the saddle-point problem

$$\min_{\lambda\geq 0} \max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] - \lambda\left(\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right] - \beta\right). \tag{3}$$

3.2 The Successor Measure and Forward-Backward Representations

The *successor representation* (Dayan, 1993), and more generally, the successor *measure* (Blier et al., 2021) are objects that encode the value function for any reward function associated to a given policy in an MDP. Let $\rho \in \mathscr{P}(\mathcal{S})$ denote a probability measure on the state space, generally interpreted as, say, a dataset. For any policy π, its successor measure $M^\pi : \mathcal{S} \times \mathcal{A} \to \mathscr{P}(\mathcal{S})$ is a (conditional) measure expressed relative to the measure ρ, via

$$M^\pi(s, a, \mathrm{d}s') = \mathbb{E}_\pi \left[\sum_{t\geq 0} \gamma^t \delta_{s'}(S_t) \,\middle|\, S_0 = s, A_0 = a\right] \rho(\mathrm{d}s'),$$

where $\delta_{s'}$ is the Dirac measure which places all mass on the state s'. Crucially, denoting by Q^π_r the action-value function for policy π on the reward function $r : \mathcal{S} \to \mathbb{R}$, we have that

$$Q^\pi_r(s, a) = (M^\pi r)(s, a) := \int r(s') M^\pi(s, a, \mathrm{d}s').$$

That is, M^π is a linear operator from reward functions to value functions, permitting *zero-shot policy evaluation* (Dayan, 1993; Blier et al., 2021). To circumvent the policy-dependent nature of the successor measure and permit zero shot policy *optimization*, Touati & Ollivier (2021) introduces a factorization which identifies each reward function with a particular optimal policy. Under their *forward-backward* (FB) representation, for policies π_z parameterized by a latent $z \in \mathbb{R}^d$, the successor measure is expressed by

$$M^{\pi_z}(s, a, \mathrm{d}s') = F(s_0, a_0, z)^\top B(s')\rho(\mathrm{d}s'), \quad \text{and} \quad \pi_z(a \mid s) := \operatorname*{argmax}_a F(s, a, z)^\top z, \tag{4}$$

where $F : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}^d$ and $B : \mathcal{S} \to \mathbb{R}^d$. The cleverness of this representation lies in the identity that for any policy π and reward function r, $(M^\pi r)(s, a) \equiv \int r(s')M^\pi(s, a, \mathrm{d}s') = Q^\pi(s, a)$. As a consequence, taking $z^r := Br \equiv \int B(s)r(s)\,\mathrm{d}\rho(s)$, we have that $F(s, a, z^r)^\top z^r = Q^{\pi_{z^r}}(s, a)$, so that π_{z^r} is optimal for the reward r by (4), as it is greedy with respect to its action-value function for r.

Underlying this idea is the following concept that we use throughout. The FB representation learns a *family* of successor measures parameterized by $z \in \mathbb{R}^d$. By the identity that π_{z^r} is optimal for rewards satisfying $z^r = Br$, the latents z^r simultaneously describe both reward functions and policies. More precisely, an embedding z describes a particular *behaviour*—it induces a policy which is optimal for an equivalence class of rewards. *Prompting* an FB representation with z elicits this behaviour.

4 Zero-Shot Constrained Policy Optimization

In order to treat zero-shot constrained policy optimization, we take the Lagrangian formulation of performing bilevel optimization on the returns from a modified reward function of the form $r - \lambda c$, where r is the nominal reward, c is the cost function, and $\lambda \geq 0$ is the dual variable to be optimized along with the policy. The key insight is that for any given λ, a pretrained FB representation could, in principle, compute the corresponding optimal $\pi_{z^{r-\lambda c}}$ directly—this removes one layer of optimization.

Let us now illustrate this more precisely. Suppose $\beta = 0$. As discussed in Section 3.1, solving the constrained RL problem consists of solving the following, where S_t^π is the (random) state visited under π at time t,

$$\widehat{\lambda} = \operatorname*{argmin}_{\lambda \geq 0} \max_\pi \mathbb{E}_{S_0 \sim p_0}\left[\mathbb{E}\left[\sum_{t \geq 0} \gamma^t \underbrace{(r(S_t^\pi) - \lambda c(S_t^\pi))}_{r^\lambda(S_t^\pi)}\right]\right]. \tag{5}$$

Notably, within the outer expectation, we simply have an expression for the value function for π corresponding to the reward function r^λ. Given an FB representation satisfying (4), we can simplify (5) by directly performing policy optimization,

$$\widehat{\lambda} = \operatorname*{argmin}_{\lambda \geq 0} \mathbb{E}_{S_0 \sim \rho_0}\left[\max_a F(S_0, a, z^{r^\lambda})^\top z^{r^\lambda}\right]. \tag{6}$$

Our main result is a method for optimizing (6) at inference time. The following theorem certifies an optimization scheme for efficiently learning $\widehat{\lambda}$ and the constrained optimal policy.

Theorem 1 *Let (F, B) denote an FB representation satisfying Equation (4). Suppose for cost c there exists a finite Λ_c such that $V^\pi - \Lambda_c C^\pi$ is bounded for at least one constraint-satisfying policy π (i.e., with probability 1, π does not exceed the cost budget). For any $\lambda_0 \geq 0$ and $N_{\mathsf{sample}} \in \mathbb{N}$, consider the iterates $\{\lambda_k\}_{k \geq 0}$ given by*

$$\lambda_{k+1} = \lambda_k + \eta_k \frac{1}{N_{\mathsf{sample}}} \sum_{i=1}^{N_{\mathsf{sample}}} \max_a F(S_i, a, z^{r - \lambda_k c})^\top z^c, \quad S_i \overset{\text{iid}}{\sim} \rho_0, \tag{7}$$

where $\{\eta_k\}_{k\geq 0}$ is a sequence of step sizes satisfying the Robbins-Munro conditions (Robbins, 1951). Then $\lambda_k \to \hat{\hat{\lambda}}$ with probability 1, such that $\pi_{z^{r-\hat{\lambda}c}}$ is optimal for the constrained RL problem with cost c and budget 0. *[Proof]*

Theorem 1 provides us with an inference-time (one-dimensional) stochastic gradient descent scheme for recovering the optimal Lagrange multiplier, enabling zero-shot constrained policy optimization via the FB representation. We refer to this as an "inference-time" scheme because, given a pretrained FB representation, no interaction is required for policy optimization: one must merely solve a simple convex optimization problem in one dimension. The FB representation does the heavy lifting, allowing us to substitute the inner policy optimization from (5), which would normally require expensive RL training, with an exact expression for the optimal value and cost functions.

4.1 Mitigating FB estimation errors

In order to truly perform zero-shot offline constrained policy optimization, our method relies on predictions of the long-term cost from the FB representation under predicted optimal policies $\pi_{z^{r-\lambda c}}$ across a wide range of λ. It is imperative that these predictions are accurate—overestimation of the long-term cost can induce policies that are much too conservative, precluding any reward optimization, and underestimation of the long-term cost can induce policies that simply exceed their cost budget. A crucial challenge in our setting is that, since some r, c pairs may necessitate arbitrarily large λ, the latents $z^{r-\lambda c}$ can easily veer outside the distribution over latents seen during pretraining, hindering prediction quality.

Henceforth, let $\hat{C}^{r,c}_{\lambda}$ denote the zero-shot unbiased (e.g., Monte Carlo) *estimate* of the expected cost returns of policy $z^{r-\lambda c}$ on cost function c,

$$\mathbb{E}[\hat{C}^{r,c}_{\lambda}] := \mathbb{E}_{s_0\sim p_0}\left[F(s_0, a^{\star}_{s}, z^{r-\lambda c})^{\top} z^{c}\right] \quad \text{where } a^{\star}_{s} := \operatorname*{argmax}_{a} F(s, a, z^{r-\lambda c})^{\top} z^{r-\lambda c}. \tag{8}$$

In the remainder of this section, we present methods for overcoming both overestimation and underestimation in $\hat{C}^{r,c}_{\lambda}$, enabling reliable zero-shot constrained policy optimization.

Overestimation From Equation (7), it is clear that for a budget $\beta = 0$, the optimization will result in λ continually increasing until the estimated costs under the FB representation are less than or equal to zero, since the gradient of λ will always be negative as long as $\hat{C}^{r,c}_{\lambda} > 0$. If such a result is infeasible, or infeasible under the learnt F and B representations (overestimation), this will result in λ continuing to increase, potentially degrading reward performance without significantly improving constraint satisfaction. Importantly, λ will continue to increase even if $\hat{C}^{r,c}_{\lambda}$ is constant or even increasing, as can be seen for example in Figure 1

We propose two modifications. First, we introduce a regularizer on λ to discourage unbounded growth in λ when $\hat{C}^{r,c}_{\lambda}$ is no longer decreasing significantly. Particularly, we add a penalty $\mathcal{R}(\lambda) = \lambda^2$ to (7) to help mitigate the issue.

Second, we propose a post-optimization procedure using this zero-shot estimate of the cost under the policy induced by each λ_n obtained during the gradient descent scheme in Equation 7. In particular, we store the estimate $C_n = \hat{C}^{r,c}_{\lambda_n}$ for each λ_n encountered during gradient descent. After optimization has terminated, we select the minimum λ_n for which C_n is within a factor of $\alpha \in [0, 1]$ from the minimum, relative to the range of costs across all iterates.

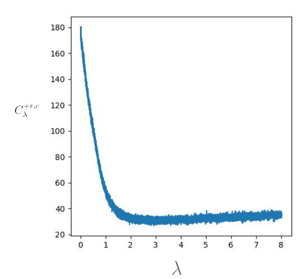


Figure 1: Example of instance where estimated cost does not fall below zero as λ continues to increase.

Underestimation On the other hand, when estimating the long-term cost of the policy induced by $z^{r-\lambda c}$ using (8), we find that

[1]Depending on the application, the initial state distribution may be known (for example, given a known single initial state, the Dirac delta distribution may be used. If it is unknown, it is reasonable to use the FB training dataset as a proxy.

Algorithm 1 Zero-Shot Constrained RL

Require: FB representation networks F, B, dataset D and the initial state distribution ρ_0[1]
Require: Initial $\lambda_0 \geq 0$, learning rate η, batch size N_{batch}, number of steps N_{grad}, cost threshold α

Sample minibatch $(s_1, \ldots, s_n) \overset{\text{iid}}{\sim} D$
$z^r \leftarrow \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} B(s_i) r(s_i)$
$z^c \leftarrow \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} B(s_i) c(s_i)$
for $n \in \{1, \ldots, N_{\text{grad}}\}$ **do**
 $z \leftarrow z^r - \lambda_n z^c$
 $z \leftarrow \sqrt{d} \frac{z}{\|z\|}$ ▷ Normalize embedding
 Sample minibatch $\mathsf{S} = (s_1, s_2, \ldots, s_{N_{\text{batch}}}) \overset{\text{iid}}{\sim} \rho_0$
 $a_i^* \leftarrow \text{argmax}_a F(s_i, a, z)^\top z$ for each $i \in \{1, \ldots, N_{\text{batch}}\}$
 $\mathcal{L}(\lambda_n, \mathsf{S}) = -\lambda_n \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \texttt{ReLU}\left(\texttt{stop-grad}(F(s_i, a_i^*, z))^\top z^c\right) + \mathcal{R}(\lambda_n)$
 $C_n \leftarrow \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \texttt{ReLU}(F(s_i, a_i^*, z)^\top z^c)$
 $\lambda_{n+1} \leftarrow \lambda_n - \eta \nabla_\lambda \mathcal{L}(\lambda_n, \mathsf{S})$
end for
$\lambda = \min_n \lambda_n : C_n \leq \alpha(C^{\max} - C^{\min}) + C^{\min}$ where $C^{\max} = \max_n C_n$ and $C^{\min} = \min_n C_n$
return λ (**Relaxed Min-Cost**), λ_N (**Last Iterate**)

the FB representation can also exploit *negative* cost predictions, resulting in poor behaviour. In principle, given that $c \geq 0$, a perfect FB representation would not predict negative long-term costs, however, due to small training error, this cannot be avoided conclusively. As such, when estimating the long-term cost of our policy, we instead follow (9), with $\texttt{ReLU}(x) := \max(0, x)$,

$$\hat{C}_\lambda^{+,r,c} := \mathbb{E}_{s \sim p_0}\left[\texttt{ReLU}(F(s, a_s^\star, z^{r-\lambda c})^\top z_c)\right] \quad \text{where } a_s^\star := \underset{a}{\text{argmax}}\, F(s, a, z^{r-\lambda c}). \tag{9}$$

With the addition of these modifications, we arrive at the proposed Algorithm 1.

5 Experimental Results

In this section, we investigate the performance of our proposed approach on some illustrative constrained environments. Particularly, our experiments are designed to illustrate whether our approach masterfully navigates the Pareto front between success rate with respect to the nominal reward r and constraint violation with respect to c. We evaluate our method in two environments adapted from (Touati & Ollivier, 2021), discrete and a continuous state-space grid worlds, both with discrete action spaces. While the original environments in (Touati & Ollivier, 2021) had internal walls, in our experiments, we pretrain an FB representation in an empty grid and represent the walls via constrained states. In a sense, constrained policy optimization can generalize *across transition dynamics*.

FB Training We train FB representations across three seeds following the method from Touati & Ollivier (2021), though we update the forward network architecture to match that from Touati et al. (2022). As in Touati & Ollivier (2021), we replace the argmax in Equation (4) with a softmax in the training updates, though we lower the temperature τ to 50 in the discrete environment and 100 in the continuous environment. All hyperparameters for FB training as well as training curves are reported in Appendix B. As can be seen in Figure 5, all learned FB representations converge to near 100% success on unconstrained goal-reaching tasks.

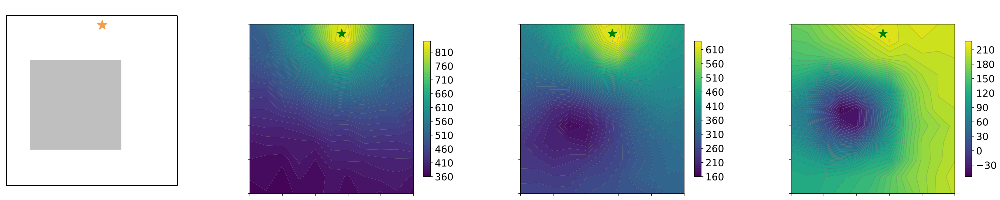
Evaluation Setup We evaluate each pre-trained FB representation according to the following setup. We first generate a random environment by sampling random constraints where a single constraint consists of a contiguous rectangular section of the grid at which the agent receives a cost of 1, and then sampling a random goal non-overlapping with the constraints, at which the agent receives a reward of 1. We provide an example of such a generated environment in Figure 2a. Exact details of environment generation can be found in Appendix C.1.

For each environment, we estimate z^r and z^c given by $z^\bullet = \mathbb{E}_{S\sim\rho}[\bullet(s)B(s)]$ for $\bullet \in \{r, c\}$ via Monte Carlo, and execute Algorithm 1 (see Appendix C.2 for hyperparameters) to approximate $\widehat{\lambda}$. Finally, we test policy $\pi_{z^{r-\lambda c}}$ for 100 rollouts with random initial states and report the average success and violation rate across all rollouts. We repeat this 50 times across each FB representation.

We evaluate our methods **Relaxed Min-Cost**, which uses the additional procedure in Algorithm 1 to select a less conservative λ, and **Last Iterate**, which selects the final λ after iterating Equation 7 for a fixed 10,000 steps. We compare against **Privileged**, a method that provides an upper bound on our methods' possible performance, by sweeping over λ and using online evaluations to select the best value. To select the best λ for the **Privileged** method, we use a 5% quantile threshold on the cost and select the λ with the highest success rate with costs below the threshold. We emphasize that unlike **Privileged**, our methods do not have access to online evaluations when selecting λ; we use the **Privileged** method solely as an approximation of the best achievable algorithm performance under a given pretrained FB representation.

5.1 Results

To qualitatively demonstrate the importance of selecting a good λ for constrained policy optimization with FB, Figure 2 illustrates various value functions obtained by varying λ in an environment with a fixed goal and constraint. As can be seen in Figure 2b, selecting a λ which is too low results in a value function that does not reflect the constraint. However, selecting a λ that is too high is also detrimental, as the value function loses distinction of the goal's location (Figure 2d). Using Algorithm 1 to the select λ (Figure 2c) results in a good balance between the constraint and the goal.

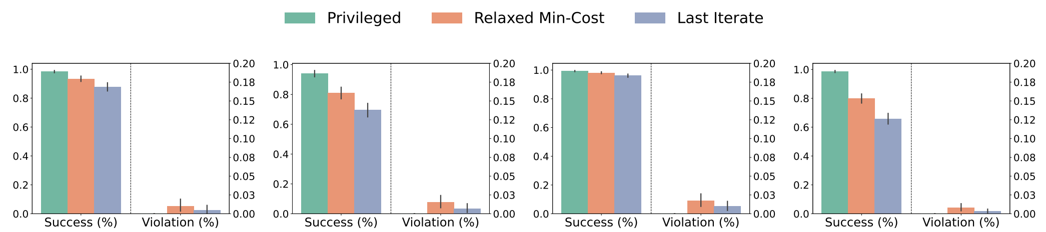


(a) Environment (b) $V^{*,r-\lambda c} : \lambda < \widehat{\lambda}$ (c) $V^{*,r-\lambda c} : \lambda = \widehat{\lambda}$ (d) $V^{*,r-\lambda c} : \lambda > \widehat{\lambda}$

Figure 2: (a) Continuous grid-world environment with randomly generated constraint (grey) and goal (star) (b-d) corresponding optimal value function $V^{*,r-\lambda c}(s) = \max_a F(s, a, z^{r-\lambda c})^T z^{r-\lambda c}$ under various λ: (c) optimal λ obtained using Algorithm 1 (b) λ less than optimal and (d) λ larger than optimal. Qualitatively, the scale of λ has a clear impact on the quality of the value function.

Quantitatively, we evaluate the average performance of our method across all seeds and evaluation runs. Figure 3 compares the average performance of our methods **Relaxed Min-Cost** and **Last Iterate** in each environment {discrete, continuous} with different number of constraints {one, two} versus the **Privileged** performance. In all cases, we find that our method is able to achieve very low constraint violations, particularly the **Last Iterate** variant. Additionally, in the single constraint cases in both discrete and continuous environments, there is also very little degradation in goal-reaching success versus the **Privileged** method. In the two constraint environments, there is a more significant degradation, however, we still obtain reasonable goal reaching success in both environments, particularly when using the **Relaxed Min-Cost** variant.

The **Relaxed Min-Cost** variant can be interpreted as a method for shifting the solution along the Pareto-optimality curve. In Figure 4, we examine this hypothesis by considering the performance of our method against the Pareto-optimal curve for a fixed λ. We aggregate the performance in all



(a) Discrete/One (b) Discrete/Two (c) Continuous/One (d) Continuous/Two

Figure 3: Goal-reaching success (%) and violation rate (%) for our method **Relaxed Min-Cost** and **Last Iterate** versus **Privileged** on {discrete, continuous} gridworlds with {one, two} constraints.

environments and constraint settings while holding λ constant to produce the Pareto-optimal curve for performance that can be achieved without adjusting λ on a per-environment basis.

In Figure 4, we can see that both the **Relaxed Min-Cost** and **Last Iterate** methods lie very close to fixed-λ Pareto-optimality curve, but at different points on the curve. The **Last Iterate** method selects a point on the curve where near-zero constraint violations can be obtained with a minimal amount of degradation in goal-reaching success. The **Relaxed Min-Cost** variant relaxes the threshold on constraint violations slightly in order to achieve higher goal-reaching success, but remains on the Pareto-optimality curve.

Notably, **Privileged** performance is able to achieve a result above the Pareto-optimality curve for a fixed λ. This is achievable since the fixed-λ method is not able to vary λ across environments depending on various conditions, while the **Privileged** method is able to do so. While our method does achieve good results on average, it is not precise enough in individual environments to match the **Privileged** performance, suggesting areas for future development.

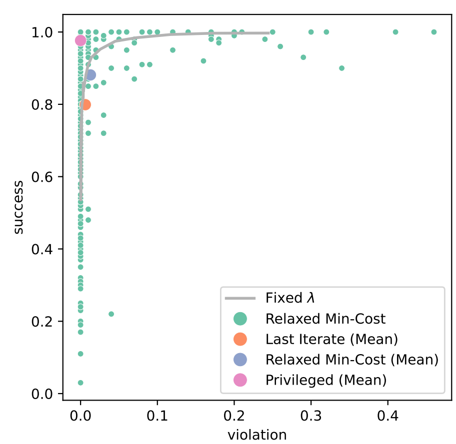


Figure 4: Pareto-optimality curve for fixed-λ using average success and violation rates across all evaluations (continuous and discrete, one and two constraints) with a fixed λ. Per-environment performance of our method **Relaxed Min-Cost** is compared against the curve as well as average performance of all methods **Relaxed Min-Cost**, **Last Iterate** and **Privileged**.

6 Conclusions

Overall, we find that our proposed method is able to perform constrained policy optimization in a zero-shot manner, effectively balancing reward and cost terms to produce a policy representation that is adherent to constraints while remaining generally successful at the goal-reaching tasks.

Our results also suggest many promising avenues for future work. For one, we find that there is considerable variance in the performance of our method across different environment configuration. Incorporating methods for reducing noise in model-based RL, such as ensembling, could be a promising avenue for improving inference-time performance. Moreover, while our proposed method is restricted to inference-time adaptation, it is possible that adapting the FB training procedure itself to specifically support the constrained policy optimization case could enhance our performance.

A Proof of Theorem 1

For any policy π and $\lambda \in \mathbb{R}_+$, define

$$f(\pi, \lambda) := \mathop{\mathbb{E}}_{S\sim\rho_0} \left[V^\pi(S) - \lambda C^\pi(S)\right],$$

where $C^\pi(s) := \mathbb{E}\sum_{t\geq 0}\gamma^t c(S_t^\pi)$, and where S_t^π is the (random) state visited under the policy π at time t. Crucially, we note that C^π is simply the value function corresponding to the MDP with reward function c, and more generally, $V^\pi - \lambda C^\pi$ for any $\lambda \in \mathbb{R}$ is the value function corresponding to the MDP with reward function $r - \lambda c$.

Next, define $\overline{f}(\lambda) = \max_\pi f(\pi, \lambda)$. Recall that a solution $\pi^\star$ to the constrained RL problem satisfies

$$\min_{\lambda\geq 0} f(\pi^\star, \lambda) = \min_{\lambda\geq 0} \overline{f}(\lambda), \tag{10}$$

as discussed in Section 3.1, when the budget $\beta = 0$. Note that for any policy π, $f(\pi, \cdot)$ is linear. Thus, $\overline{f}$, as a pointwise maximum of convex (linear) functions, is a convex function. Moreover, by the stated assumptions, we have that $\min_{\lambda\geq 0} f(\pi, \lambda) \geq M > -\infty$ (where M is an arbitrary constant) for some constraint-satisfying policy π, so that $\min_{\lambda\geq 0}\overline{f}(\lambda) = \min_{\lambda\geq 0} f(\pi^{r-\lambda c}, \lambda)$ is bounded, and the minimum is achieved at some $\lambda \leq \Lambda_c < \infty$. As a consequence, for any $\lambda_0 \in \mathbb{R}_+$, and any unbiased estimator $\widehat{C}_k$ of $\frac{\mathrm{d}}{\mathrm{d}\lambda}\overline{f}(\lambda)$ bounded with probability 1, the iterates

$$\lambda_{k+1} := \lambda_k - \eta_k \widehat{C}_k \tag{11}$$

converge to some $\widehat{\lambda} \in \mathbb{R}_+$ by the stochastic approximation theory of Robbins and Munro (Robbins, 1951). Moreover, by (4), we have that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\lambda}\overline{f}(\lambda) &= \frac{\mathrm{d}}{\mathrm{d}\lambda}\left[\max_\pi f(\pi,\lambda)\right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\lambda}\left[\max_\pi \mathop{\mathbb{E}}_{S\sim\rho_0}(V^\pi(S) - \lambda C^\pi(S))\right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\lambda}\left[\mathop{\mathbb{E}}_{S\sim\rho_0}(V^{\pi^\star}(S) - \lambda C^{\pi^\star}(S))\right]_{\pi^\star = \pi_{z^{r-\lambda c}}} && \text{By Equation (4)} \\
&= -\frac{\mathrm{d}}{\mathrm{d}\lambda}\left[\mathop{\mathbb{E}}_{S\sim\rho_0}\lambda C^{\pi^\star}(S)\right]_{\pi^\star = \pi_{z^{r-\lambda c}}} \\
&= -\mathop{\mathbb{E}}_{S\sim\rho_0}\left[\max_a F(S, a, z^{r-\lambda c})^\top z^c\right]. && \text{By Equation (4)}
\end{aligned}$$

Since $\pi_{z^{r-\lambda_c}}$ is a constraint-satisfying policy, the Monte-Carlo estimates given by

$$\widehat{C}_k = \frac{1}{N_{\mathsf{sample}}}\sum_{i=1}^{N_{\mathsf{sample}}} \max_a F(S_i, a, z^{r-\lambda_k c})^\top z^c, \quad S_i \overset{\text{iid}}{\sim} \rho_0$$

are unbiased and bounded with probability 1. Thus, substituting into the iterates of (11), we yield the iterates shown in the statement of Theorem 1. Therefore, these iterates converge with probability 1 to $\widehat{\lambda} = \min_{\lambda\geq 0}\overline{f}(\lambda)$. Then, by the defining property of the FB representation (4), it holds that $\pi_{z^{r-\widehat{\lambda}c}}$ maximizes $f(\cdot, \widehat{\lambda})$, so that $\pi_{z^{r-\widehat{\lambda}c}}$ is optimal for the constrained RL problem with budget $\beta = 0$ by (10).

B FB Training Details

For FB training, we generally follow the code provided by the authors of Touati & Ollivier (2021) (`https://github.com/ahmed-touati/controllable_agent`), but we use the forward network architecture from Touati et al. (2022) as well as the normalization of the output of

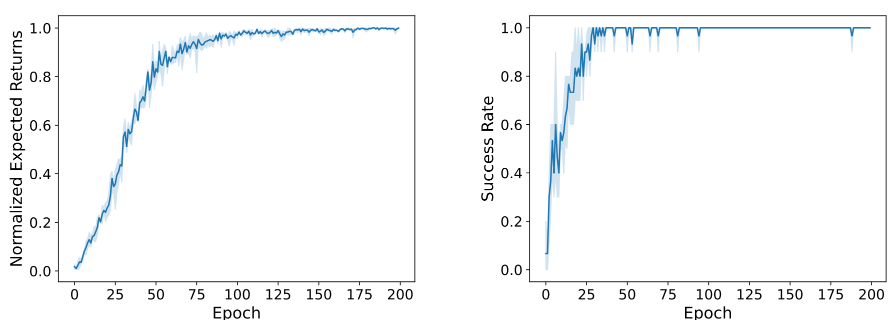
Hyperparameter	Discrete	Continuous
Epochs	200	200
Cycles per epoch	25	25
Episodes per cycle	-	4
Timesteps per episode	100	100
Updates per cycle	40	40
Exploration ϵ	-	1
Evaluation ϵ	Boltzmann $\tau = 1$	0.02
temperature τ	50	100
Learning Rate	0.0005	0.00001
Mini-batch size	128	128
Regularization coefficient	1	1
Polyak α	0.99	0.95
Discount γ	0.99	0.99
Replay buffer size	-	10^6
z dimension	100	100

Table 1: Hyperparameters used for FB training in discrete and continuous environments.

the backward network (scaled to have $\sqrt{d}$ norm). For the discrete environment, we pre-collect a full coverage dataset for training by sampling all state and action transitions in the MDP. For the continuous environment, we use rollouts with a random policy, as is used in Touati & Ollivier (2021). We use a random start state for rollouts which is at least 0.15 units from the wall.

All hyperparameters used for FB training in the continuous and discrete environments are provided in Table 1

FB training converges to very strong goal-reaching success across all seeds in both the discrete and continuous environments, as can be seen in Figure 5.



(a) Discrete (b) Continuous

Figure 5: FB training curves where performance is evaluated by goal-reaching success for all (discrete) or randomly sampled (continuous) goals. Following Touati & Ollivier (2021) we report goal-reaching success, in the discrete environment, as the expert-normalized expected returns, computed exactly using the known transition dynamics of the discrete MDP and, in the continuous environment, as average goal-reaching success over 10 test rollouts. Error bars display 95% confidence intervals across three seeds.

C Experiment Details

C.1 Environment Generation

The discrete gridworld is a 10x10 grid surrounded by walls. States are one-hot vectors indicating the position of the agent. In the continuous gridworld, the state consists of the (x, y) position of the agent, where $(x, y) \in [[0, 1], [0, 1]]$. Generating an environment consists of first generating one or two constraints and then generating a goal which does not overlap with the constraints. We constrain this generation procedure according to the limits specified in Table 2.

Environment	Discrete		Continuous	
Num Constraints	One	Two	One	Two
Constraint min width	1	1	0.1	0.1
Constraint max width	6	6	0.6	0.6
Constraint min height	1	1	0.1	0.1
Constraint max height	6	6	0.6	0.6
Constraint min area	10	1	0.1	0.01
Constraint max area	40	16	0.4	0.16
Min distance between wall and constraint	1	1	0.1	0.1
Min distance between constraints	-	3	-	0.3
Min distance between goal and constraints	2	2	0.2	0.2
Min distance between start and constraints	0	0	0.05	0.05

Table 2: Parameters used for generating constrained environments for evaluation.

C.2 Hyperparameters

We use the following hyperparameters in the instantiation of Algorithm 1 in both the discrete and continuous environments.

Hyperparameter	
λ_0	0.01
η	10^{-3}
N_{grad}	10000
N_{batch}	528
α	0.05

Table 3: Hyperparameters used to instantiate Algorithm 1

References

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

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The information geometry of unsupervised reinforcement learning. *arXiv preprint arXiv:2110.02719*, 2021.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. *Advances in neural information processing systems*, 33: 1724–1735, 2020.

Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. *arXiv preprint arXiv:2204.08957*, 2022.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International Conference on Machine Learning*, pp. 21611–21630. PMLR, 2023.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22: 400–407, 1951. URL https://api.semanticscholar.org/CorpusID:16945044.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International conference on machine learning*, pp. 9870–9879. PMLR, 2021.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? *arXiv preprint arXiv:2209.14935*, 2022.

Harley Wiltzer, Jesse Farebrother, Arthur Gretton, Yunhao Tang, André Barreto, Will Dabney, Marc G Bellemare, and Mark Rowland. A distributional analogue to the successor representation. In *International Conference on Machine Learning*, 2024.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pp. 11480–11491. PMLR, 2021.

Yihang Yao, Zuxin Liu, Zhepeng Cen, Jiacheng Zhu, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constraint-conditioned policy optimization for versatile safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36:12555–12568, 2023.

Yiming Zhang, Quan Vuong, and Keith Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.