Robust Policy Optimisation under Reward Misspecification via Support-Bounded Uncertainty

Adhya Rajaram
adhyarajaram789@gmail.com

Abstract

We propose a framework for safe policy improvement when the reward function is only a partially trustworthy proxy for the true objective. Safe Policy Improvement with Baseline Bootstrapping (SPIBB) constrains policy updates in poorly-covered regions, but assumes the proxy reward is accurate within its trusted support. We address its complementary case, bounded reward uncertainty within that support. Given a baseline π_0 and trusted support $K = \{(s, a) : \mu_{\pi_0}(s, a) \geq \tau\}$, we define a support-bounded uncertainty set $\mathcal{U}_K$ and bootstrapped policy class Π_K and prove that the worst-case safety constraint reduces exactly to a penalised occupancy-shift condition: $\Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}}\|\Delta\mu_K\|_1 \geq 0$. Lagrangian relaxation then converts this to tractable modified policy iteration. Experiments across 40 random MDP configurations show naive proxy optimisation causes Goodhart failure in 87.5% of cases; our method recovers in 75% of those with a 90% reduction in Goodhart gap. The framework is formally certifiable for misspecification radii $\varepsilon_{\text{in}} \leq 0.15$ and generalises without modification to Gymnasium environments, where naive methods catastrophically fail.

1 Introduction

A key challenge in deploying reinforcement learning systems is that reward functions are typically imperfect proxies for the true objective [2, 14]. This has two important implications for AI Safety; firstly, specifying what a system should do requires the reward to accurately represent the objective [11, 7], and secondly, any safety guarantee is only as strong as the reward function being certified against.In practice, reward functions are not directly observable, but must be inferred, hand-designed, or learned from proxy signals [16]. These may deviate substantially from the true objective in regions the designer did not anticipate. Optimising an imperfect proxy too aggressively can degrade true performance, a failure formalised by Goodhart's Law and studied empirically in reward model overoptimisation [3, 6]. A natural response is to constrain the agent's deviation from a known baseline policy in insufficiently explored regions, such as in Safe Policy Improvement with Baseline Bootstrapping (SPIBB, [10]). SPIBB bootstraps the trained policy with the baseline in poorly-covered state-action regions. However, it places uncertainty on transition model coverage, rather the reward function itself, by assuming the proxy reward is accurate wherever the baseline provides sufficient data.

We address the complementary case of this, by assuming the proxy reward R' is accurate up to a tolerance of ε_{in} on a *trusted support set* $K = \{(s, a) : \mu_{\pi_0}(s, a) \geq \tau\}$, the region well-covered by the baseline, and may differ arbitrarily outside it. This models the realistic scenario where a reward designer can vouch for the proxy in familiar regions but not elsewhere in the environment. We derive exact conditions for robust safe improvement in this setting, solve the resulting constrained optimisation via Lagrangian relaxation, and validate the framework empirically across custom and standard Gymnasium environments.

2 Related Work

Safe policy improvement: Safe Policy Improvement with Baseline Bootstrapping (SPIBB, [10]) and its extension Soft-SPIBB [12] constrain policy updates to state-action pairs with sufficient empirical coverage, bootstrapping to the baseline elsewhere to provide high-probability improvement guarantees. Both methods address uncertainty in the transition model that arises from limited data and assume the reward function is accurately observed wherever there is sufficient coverage. We address the case where the exact transition model is known (like model-based RL), but the reward function is only partially trustworthy within its coverage region. Our bootstrapped policy class Π_K is identical in structure to SPIBB's safe set, but our safety constraint is derived from reward uncertainty instead of the sampling error.

Reward misspecification and Goodhart's Law: The degradation of true performance under proxy reward optimisation is well-documented empirically [3, 6] and theoretically [14]. KL regularisation is commonly used as a mitigation for this, but Kwa et al. [9] show that it fails to provide guarantees under heavy-tailed misspecification. Our penalty term $\varepsilon_{\text{in}}\|\Delta\mu_K\|_1$ is derived from the adversarial worst case over $\mathcal{U}_K$ and is not chosen heuristically. STARC metrics [17] provide pseudometrics on reward function space that give tight upper and lower bounds on worst-case regret. Our ε_{in} can be interpreted as an ℓ_∞ ball radius in this space, restricted to K. Skalse and Abate [18] show that IRL is highly sensitive to behavioural model misspecification, motivating why R' cannot be trusted globally, but only within K.

Conservative offline reinforcement learning. Conservative Q-Learning [8] penalises out-of-distribution actions via a Q-value regularisation term, addressing the distributional shift in offline settings. Our Lagrangian penalty $(1 + \lambda)R' - \lambda\varepsilon_{\text{in}}v$ is structurally similar, but applied to occupancy-measure shifts in the reward-uncertainty direction rather than the action-distribution shift. The connection between the two suggests our framework may extend to function-approximation settings.

Robust Markov decision processes. Iyengar [5] and **(author?)** [13] study MDPs with uncertainty sets over *transition kernels*, providing minimax-optimal policies through rectangular uncertainty sets. We introduce the reward-robust analogue: an ℓ_∞ uncertainty set over reward functions restricted to the trusted support. Unlike transition-robust MDPs, our uncertainty set is non-rectangular in the full reward space (it acts only on K), which is what enables the closed-form reduction in Theorem 1.

3 Problem Formulation

3.1 Markov Decision Processes and Occupancy Measures

We work with a discounted MDP $M = (S, A, T, \gamma)$, where S is a finite state space, A a finite action space, $T : S \times A \times S \to [0, 1]$ a transition kernel, and $\gamma \in (0, 1)$ a discount factor. Rather than a single reward function, we consider a set $\mathcal{U} \subset \{R : S \times A \times S \to \mathbb{R}\}$ of *plausible reward functions*, representing uncertainty about the true objective or reward misspecification.

The **discounted state-action occupancy measure** of a policy π under initial distribution d_0 is:

$$\mu_\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr_\pi(s_t = s,\, a_t = a;\, d_0) \tag{1}$$

The occupancy measure satisfies $\mu_\pi \geq 0$ and $\sum_{s,a} \mu_\pi(s, a) = 1$. Thus, it defines a probability distribution over state-action pairs. The expected discounted return then allows for the **inner product representation**:

$$J_R(\pi) = \langle \mu_\pi,\, R \rangle = \sum_{s,a} \mu_\pi(s, a) \sum_{s'} T(s'|s, a)\, R(s, a, s') \tag{2}$$

This representation reduces the policy comparison to the comparison of occupancy measures, which makes the effect of reward uncertainty more explicit.

3.2 Support-Bounded Reward Uncertainty

Trusted support: Given a baseline policy π_0 and threshold $\tau > 0$, we define the *trusted support set*:

$$K = \{(s, a) : \mu_{\pi_0}(s, a) \geq \tau\} \tag{3}$$

K contains state-action pairs visited sufficiently often under π_0 , such that the proxy reward R' can be trusted to approximate the true reward.

Uncertainty set. Given the misspecification radius $\varepsilon_{\text{in}} > 0$, we define:

$$\mathcal{U}_K = \left\{R : |R(s, a, s') - R'(s, a, s')| \leq \varepsilon_{\text{in}},\ \forall (s, a) \in K,\ \forall s' \in S\right\} \tag{4}$$

The proxy R' is assumed to be accurate within ε_{in} on transitions in K and may differ arbitrarily outside K. This is an ℓ_∞-ball in the reward function space restricted to K, similar to STARC-metric balls [17].

Bootstrapped policy class. To prevent the exploitation of reward uncertainty outside K, we restrict to:

$$\Pi_K = \{\pi : \pi(a|s) = \pi_0(a|s),\ \forall (s, a) \notin K\} \tag{5}$$

Π_K restricts the policy to π_0 on untrusted regions. **Note that this constraint must be enforced in the policy optimisation algorithm**, not simply assumed in the theoretical derivation (see Section 6 for more details).

3.3 Robust Baseline Improvement Objective

We seek to find a policy $\pi^* \in \arg\max_{\pi \in \Pi_K} J_{R'}(\pi)$ subject to the *robust baseline-improvement constraint*:

$$\min_{R \in \mathcal{U}_K} J_R(\pi) \;\geq\; \min_{R \in \mathcal{U}_K} J_R(\pi_0) \tag{6}$$

The new policy must not be worse than π_0 for every reward function in $\mathcal{U}_K$, not just for the proxy R'. We also set $\Delta\mu = \mu_\pi - \mu_{\pi_0}$ and $\Delta\mu_K = \Delta\mu \cdot \mathbf{1}_K$ (which is the component of $\Delta\mu$ restricted to K).

4 Main Result

Theorem 1 (Support-Bounded Robust Baseline Improvement)**.** *Let R', π_0, τ, ε_{in}, K, $\mathcal{U}_K$, and Π_K be as defined in Section 3. Thus, for any $\pi \in \Pi_K$:*

$$\min_{R \in \mathcal{U}_K} \big(J_R(\pi) - J_R(\pi_0)\big) \;\geq\; 0 \quad\Longleftrightarrow\quad \underbrace{\Delta\mu_K \cdot R'_K}_{\textit{proxy improvement}} \;-\; \underbrace{\varepsilon_{\text{in}} \|\Delta\mu_K\|_1}_{\textit{uncertainty penalty}} \;\geq\; 0 \tag{7}$$

, where $R'_K(s,a) = \sum_{s'} T(s'|s,a)\, R'(s,a,s')$ is the expected proxy reward at (s,a). If (7) holds, π improves on π_0 for every $R \in \mathcal{U}_K$.

Proof sketch. Since $\pi \in \Pi_K$, the bootstrapping constraint (5) forces $\Delta\mu(s,a) = 0$ for all $(s,a) \notin K$ exactly. The minimisation therefore reduces to:

$$\min_{R \in \mathcal{U}_K} \sum_{(s,a) \in K} \Delta\mu(s,a) \sum_{s'} T(s'|s,a)\, R(s,a,s')$$

For each $(s,a) \in K$, the adversary independently sets $R(s,a,s') = R'(s,a,s') - \varepsilon_{\text{in}} \cdot \operatorname{sgn}(\Delta\mu(s,a))$ for all s', minimising the contribution of that state-action pair. Since $\sum_{s'} T(s'|s,a) = 1$, this yields:

$$\Delta\mu(s,a)\big[R'_K(s,a) - \varepsilon_{\text{in}}\big] = \Delta\mu(s,a)\, R'_K(s,a) - \varepsilon_{\text{in}}\, |\Delta\mu(s,a)|$$

Summing over K gives the closed form in (7). The adversary's choice is both necessary and sufficient to achieve the minimum and vice versa. □ □

Full proof. For the full proof, see Appendix A.

Remark 1 (Necessity of Π_K enforcement)**.** *The Π_K constraint is load-bearing: without it, $\Delta\mu(s,a) \neq 0$ is possible outside K, where R is unconstrained. In that case, the minimum over $\mathcal{U}_K$ is $-\infty$ whenever the policy deviates from π_0 in untrusted regions. Theorem 1 provides no guarantee for $\pi \notin \Pi_K$. We confirm empirically that enforcing Π_K in the algorithm is necessary: without it, the robust method reduces to naive proxy optimisation (Section 6, Figure 2).*

Remark 2 (Feasibility)**.** *The right-hand side of (7) may not be achievable by any $\pi \in \Pi_K$. This occurs when the reward-hacking behaviour is localised outside K, particularly when the distractor state is never visited by π_0. In this case, the constraint is* structurally infeasible*: no Lagrangian multiplier can enforce it. We observe formal certification in $\varepsilon_{in} \leq 0.15$ and discuss this limitation in Section 7.*

5 Lagrangian Relaxation and Algorithm

Theorem 1 provides a tractable certificate, but solving the constrained problem $\max_{\pi \in \Pi_K} J_{R'}(\pi)$ subject to (7) requires searching over policies. Hence, we convert to an unconstrained Lagrangian.

5.1 Lagrangian Formulation

We introduce a dual variable $\lambda \geq 0$ and form the Lagrangian:

$$\mathcal{L}(\pi, \lambda) = J_{R'}(\pi) - \lambda\big(\varepsilon_{\text{in}} \|\Delta\mu_K\|_1 - \Delta\mu_K \cdot R'_K\big) \tag{8}$$

The dual function is $d(\lambda) = \max_{\pi \in \Pi_K} \mathcal{L}(\pi, \lambda)$ and the dual problem is $\min_{\lambda \geq 0} d(\lambda)$.

5.2 Reduction to Modified Policy Optimisation

We apply the ℓ_1–ℓ_∞ duality identity:

$$\|\Delta\mu_K\|_1 = \max_{\|v\|_\infty \leq 1} v \cdot \Delta\mu_K \tag{9}$$

Substituting (9) into (8) and exchanging $\min_\pi$ and $\max_v$ via the minimax theorem (valid since $\mathcal{L}(\pi, \lambda, v)$ is linear in μ_π for fixed v and linear in v for fixed μ_π. Over compact convex sets Π_K and $\{v : \|v\|_\infty \leq 1\}$), the inner maximisation over v yields $v^*(s, a) = \text{sgn}(\Delta\mu(s, a))$. The problem then reduces to standard policy iteration under the *modified reward*:

$$\tilde{R}^\lambda(s, a, s') = (1 + \lambda)\, R'(s, a, s') - \lambda\, \varepsilon_{\text{in}}\, v(s, a), \quad (s, a) \in K \tag{10}$$

with $v(s, a)$ updated to $\text{sgn}(\Delta\mu(s, a))$ at each iteration. The modified reward scales R' by $(1 + \lambda)$ and subtracts a penalty proportional to $\lambda\varepsilon_{\text{in}}$ in the direction of the occupancy shift.

5.3 KKT Conditions

At the optimal (π^*, λ^*), the KKT conditions are:

- **Stationarity:** π^* is optimal under $\tilde{R}^{\lambda^*}$ within Π_K.
- **Dual feasibility:** $\lambda^* \geq 0$.
- **Complementary slackness:** $\lambda^*(\Delta\mu_K^* \cdot R'_K - \varepsilon_{\text{in}} \|\Delta\mu_K^*\|_1) = 0$.

Either $\lambda^* = 0$ (if the constraint not binding, method reduces to unconstrained proxy optimisation) or the constraint is active at equality.

5.4 Algorithm

Algorithm 1 Robust Policy Improvement via Support-Bounded Lagrangian

Require: Baseline π_0, proxy R', MDP (S, A, T, γ), threshold τ, radius ε_{in}, multiplier grid $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$

Compute μ_{π_0} by solving $(I - \gamma T_{\pi_0}^{\top})\rho = (1-\gamma)d_0$ for ρ, then $\mu_{\pi_0} = \rho \cdot \pi_0$
$K \leftarrow \{(s,a) : \mu_{\pi_0}(s,a) \geq \tau\}$
for $\lambda \in \Lambda$ **do**
    Initialise $v \leftarrow \mathbf{0}$
    repeat
        $\tilde{R}^{\lambda} \leftarrow (1+\lambda)R' - \lambda\varepsilon_{\text{in}} v$ (on K; use R' outside K)
        $\pi \leftarrow \textsc{PolicyIteration}(\tilde{R}^{\lambda}, T, \gamma, \Pi_K\text{-mask}, \pi_0)$
        $\mu_{\pi} \leftarrow \textsc{ComputeOccupancy}(\pi, T, \gamma)$
        $v' \leftarrow \operatorname{sgn}(\mu_{\pi} - \mu_{\pi_0}) \cdot \mathbf{1}_K$
    until π unchanged or max iterations reached
    $c(\lambda) \leftarrow \Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}} \|\Delta\mu_K\|_1$ (constraint value)
    Store $(\pi, c(\lambda))$
end for
return feasible policy with highest $J_{R'}$ if any; else least-violating policy

Π_K enforcement in Algorithm 1. The PolicyIteration subroutine enforces Π_K by: (i) masking Q-values to $-\infty$ for actions outside K in states with at least one trusted action and (ii) hard-setting the policy to π_0 in states where there is no trusted action. Omitting this step reduces the Algorithm 1 to naive proxy optimisation (Remark 1).

6 Experiments

We evaluate the framework on a custom tabular gridworld and two standard Gymnasium environments, addressing four areas: (1) Is Π_K enforcement necessary in practice? (2) How does our method compare to SPIBB and does our formal certification hold? (3) How robust are the results across random configurations? (4) Does the framework generalise beyond the custom-designed environment? Additional experiments (e.g. hyperparameter sweeps, scaling, stochastic transitions, distractor location analysis) are reported in Appendix C.

6.1 Experimental Setup

Environment We use a 5×5 deterministic gridworld ($\gamma = 0.95$). The true reward R^* gives +1 for reaching the goal state (bottom-right corner). The proxy reward R' additionally gives +2 for reaching a distractor state, inducing deliberate Goodhart-style reward hacking. Here, a naive optimizer fixates on the distractor and abandons the goal.

Baseline policy π_0 is an ε-greedy policy ($\varepsilon = 0.6$) over the true-reward-optimal Q-function, providing broad coverage, albeit not exhaustive ($|K| = 32/100$ state-action pairs at $\tau = 0.005$).

Methods compared We evaluate four methods at each configuration: (i) **Baseline** π_0 (for a conservative reference); (ii) **Naive** proxy optimisation (unconstrained policy iteration on R'); (iii) **SPIBB** (a model-based variant, with policy iteration on R' subject to Π_K, i.e. our method at $\lambda = 0$); (iv) **Robust** (our method, a Lagrangian sweep over $\lambda \in [10^{-2}, 10^{2}]$, with Π_K enforced).

Metrics For each method, we report the true return $J_{R^*}(\pi)$, proxy return $J_{R'}(\pi)$, Goodhart gap $J_{R'}(\pi) - J_{R^*}(\pi)$, and constraint value $\Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}}\|\Delta\mu_K\|_1$ (positive $\Rightarrow$ formally certified).

6.2 Π_K Enforcement is Necessary

Figure 1 shows the effect of enforcing the bootstrapped policy class. Without Π_K, the robust method is identical to naive. True return collapses to 0.04 (a 95% degradation relative to baseline 0.83), whilst proxy return rises to 1.77. Enforcing Π_K recovers the true return to **0.86**, matching the baseline, with a zero Goodhart gap. This confirms Remark 1 and the theorem's assumption must therefore be implemented explicitly.

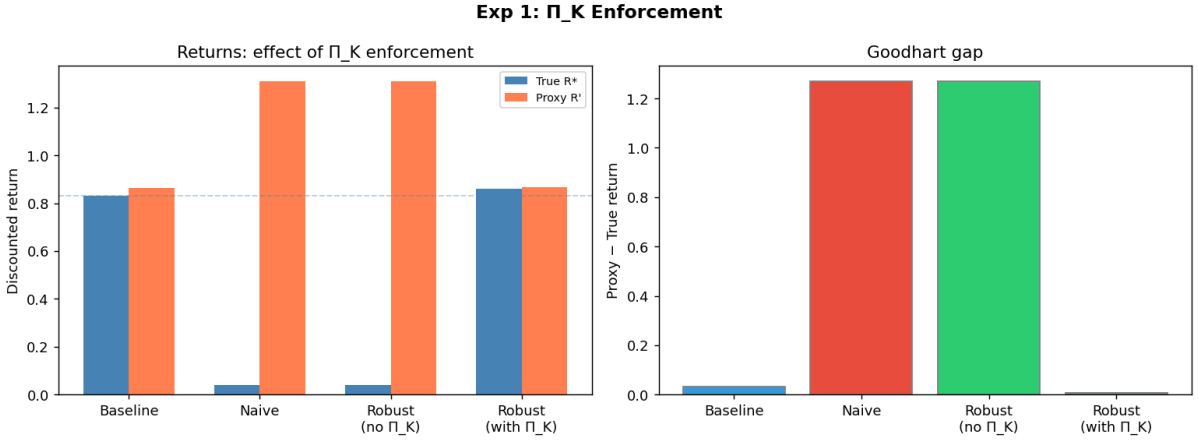

Figure 1: Effect of Π_K enforcement. True return (blue) and proxy return (orange) for each method. Without Π_K, the robust method is identical to naive and exhibits full Goodhart failure. Enforcing Π_K recovers true return to 0.86 and eliminates the Goodhart gap.

6.3 Comparison with SPIBB and Formal Certification

Figure 2 and Table 1 display the comparison with SPIBB across misspecification radii $\varepsilon_{\text{in}} \in \{0.05, 0.10, 0.15, 0.20, 0.30\}$.

Both SPIBB and the robust method achieve true return 0.86 and zero Goodhart gap across all tested ε_{in} values. This is expected, as both enforce Π_K, which is the key mechanism needed to prevent reward hacking (Section 6). The main distinction here is theoretical, as SPIBB assumes R' is exact within K and provides no guarantee under reward perturbation. Hence, our method is formally certified when the constraint is satisfied, i.e. for $\varepsilon_{\text{in}} \leq 0.15$.

At $\varepsilon_{\text{in}} > 0.15$, the constraint is not satisfied, but the empirical performance remains the same. This reflects the conservatism of the certificate rather than indicating the method

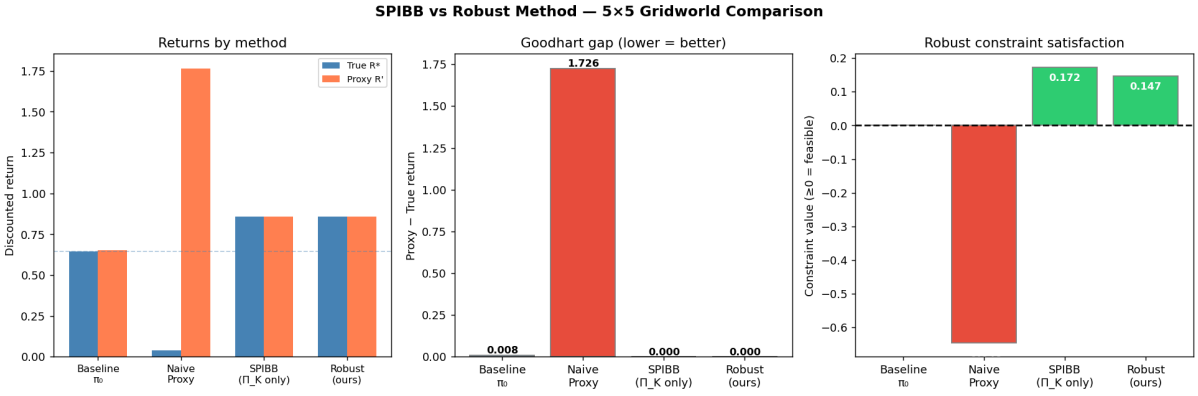

Figure 2: SPIBB vs Robust vs Naive at $\varepsilon_{\text{in}} = 0.05$, $\tau = 0.005$. *Left*: true and proxy return by method. *Right*: robust constraint value (green bars ≥ 0 indicate formal certification). Both SPIBB and Robust achieve true return 0.86 with zero Goodhart gap; only our method provides a formal reward-robustness certificate.

Table 1: Constraint values and formal certification across ε_{in}. All values recover true return 0.86 and Goodhart gap 0.

ε_{in}	SPIBB constraint	Robust constraint	SPIBB cert.	Robust cert.
0.05	+0.172	+0.147	✓	✓
0.10	+0.129	+0.079	✓	✓
0.15	+0.086	+0.012	✓	✓
0.20	+0.043	−0.056	✓	×
0.30	−0.044	−0.191	×	×

has failed. The constraint requires the proxy improvement to overcome the worst-case uncertainty penalty, which is not achievable when the misspecification budget exceeds the occupancy shift within K.

6.4 Statistical Robustness

To evaluate robustness beyond a single environment configuration, we repeat the comparison across 40 random MDPs, varying the distractor location, distractor reward magnitude (sampled from $[1.5, 3.0]$), and baseline exploration ($\varepsilon \sim \text{Uniform}[0.4, 0.8]$).

Table 2: Statistical results across 40 random MDP configurations. Mean ± standard deviation over configurations.

Metric	Baseline	Naive	Robust (ours)
True return	0.616 ± 0.081	0.142 ± 0.271	0.776 ± 0.222
Goodhart gap	0.008 ± 0.004	1.364 ± 0.652	0.143 ± 0.381
Reward-hacking rate	—	87.5%	—
Recovery rate	—	—	**75.0**%
Goodhart gap reduction	$1.36 \rightarrow 0.14$ (**90**% reduction)		

Naive optimisation causes Goodhart failure (true return < 0.3) in 87.5% of configurations. The robust method recovers (true return $>$ naive $+0.3$) in 75% of those cases, reducing the mean Goodhart gap by 90%. The 25% of non-recovered cases correspond to configurations where the distractor lies outside K, consistent with Remark 2.

6.5 Gymnasium Generalisation

To test whether the framework generalises beyond the custom gridworld, we evaluate on two discrete Gymnasium environments: **Taxi-v4** (500 states, 6 actions) and **CliffWalking-v1** (48 states, 4 actions).

In each case, we construct a proxy reward by inverting a large negative penalty (for Taxi, the -10 illegal-action penalty, and, for CliffWalking, the -100 cliff-entry penalty) into a positive reward, inducing strong Goodhart failure under naive optimisation.

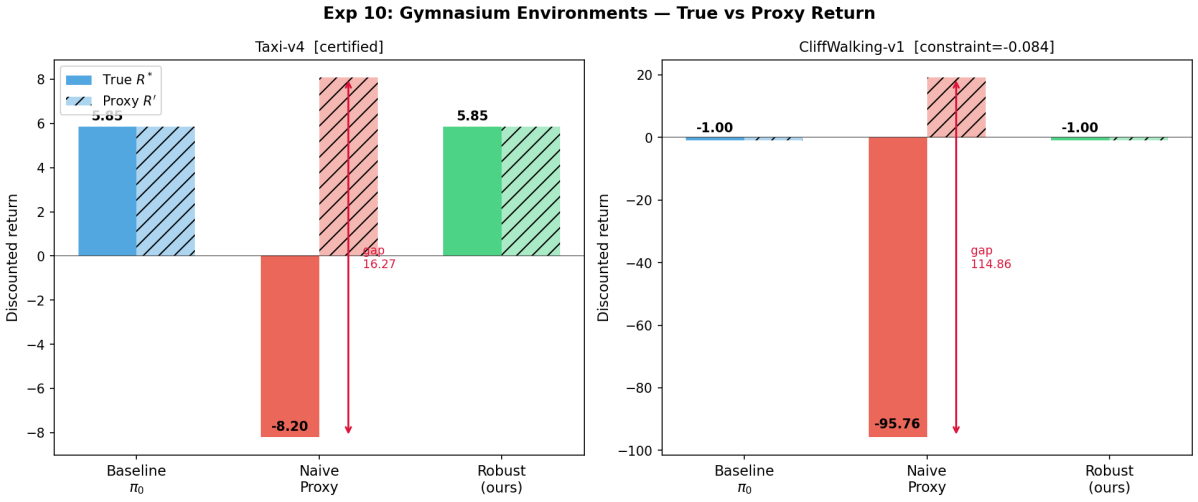

Figure 3: True return by method on Taxi-v4 and CliffWalking-v1. Naive proxy optimisation causes catastrophic Goodhart failure in both environments. The robust method recovers to baseline performance on Taxi-v4 (formally certified) and to optimal on CliffWalking-v1.

The results are shown in Figure 3. On **Taxi-v4**, the naive true return collapses to -8.20 (the agent spams illegal actions for their proxy reward); the robust method recovers to $+5.85$, matching the baseline, and is formally certified (constraint $= +0.008$). On **CliffWalking-v1**, naive true return is -95.76 (the agent loops on cliff entry); the robust method recovers to -1.00, the optimal true return, without any modification to the algorithm. These results confirm that the framework requires no environment-specific tuning and generalises to much larger state spaces.

7 Discussion

The results reveal the following findings:

The most important mechanism is Π_K enforcement The central finding is that enforcing the bootstrapped policy class Π_K in the algorithm, not the Lagrangian penalty, is what prevents Goodhart failure. Without Π_K, the robust method is identical to naive

regardless of λ (Figure 1), because the Lagrangian penalises occupancy shifts within K but cannot constrain behaviour outside it. With Π_K enforced, the agent is locked to π_0 in untrusted regions, removing the distractor from the effective action space. This has the practical implication that the theorem's assumptions must be implemented faithfully, not merely assumed.

Formal certificate vs empirical performance Our method and SPIBB achieve identical empirical performance across all tested configurations (Table 1). This is expected, as both enforce Π_K, which is the binding constraint. The Lagrangian adds a formal reward-robustness certificate in addition to this. When the constraint (7) is satisfied, the policy is provably safe under every $R \in \mathcal{U}_K$, not simply under the proxy. However, SPIBB provides no such guarantee; it assumes R' is exact within K and has no mechanism for handling reward perturbations. This distinction matters most in high-stakes settings where the proxy reward is known to be imperfect but the misspecification budget ε_{in} can be estimated.

The feasibility gap Formal certification holds for $\varepsilon_{\text{in}} \leq 0.15$ but not beyond (Table 1). At larger misspecification budgets, the uncertainty penalty $\varepsilon_{\text{in}}\|\Delta\mu_K\|_1$ dominates the proxy improvement $\Delta\mu_K \cdot R'_K$, making the constraint structurally infeasible i.e. no policy in Π_K can satisfy it. Empirical performance is not affected as the Π_K enforcement prevents reward hacking regardless of certification status. Infeasibility is greater when the distractor lies outside K entirely. The constraint is blind to occupancy shifts in untrusted regions, so Goodhart failure that occurs outside K cannot be certified away. This accounts for the 25% of random configurations where our method does not recover.

Limitations There are three main limitations to our current framework Firstly, τ and ε_{in} are treated as known hyperparameters; in practice, they must be estimated from data or domain knowledge. Secondly, the framework is restricted to tabular MDPs: Π_K enforcement via Q-masking and exact occupancy computation both require finite state-action spaces. Thirdly, the comparison baseline set is limited to SPIBB; evaluation against PPO-Lagrangian, CPO [1] and FOCOPS [19] is needed before drawing conclusions about performance relative to the broader safe-RL literature.

Future work Below are three potential extensions to this project: Estimating ε_{in} from reward annotation disagreement or Bayesian reward models would make the framework deployable without needing manual tuning. Extending Π_K enforcement to function approximation (for example, via constrained policy gradient with occupancy regularisation) would enable application to continuous control. Finally, combining reward and transition uncertainty in a unified robust CMDP (Constrained Markov Decision Process) formulation would address the full misspecification problem that motivates this work.

8 Conclusion

We have proposed a framework for safe policy improvement under support-bounded reward misspecification. By restricting reward uncertainty to the trusted support K and the policy to the bootstrapped class Π_K, we reduce the worst-case safety constraint to a closed-form penalised occupancy-shift condition, enabling efficient certification via

Lagrangian relaxation. The main practical finding is that Π_K enforcement is the most critical implementation step: without it, the framework reduces to naive proxy optimisation regardless of the Lagrangian penalty. Our method matches SPIBB empirically whilst providing a qualitatively stronger guarantee, that formal certification under every reward function in $\mathcal{U}_K$, and generalises without modification to standard Gymnasium environments. We hope this work contributes a principled approach to the reward uncertainty problem that complements existing safe policy improvement methods and opens a path toward robust policy improvement in settings where reward misspecification is unavoidable.

References

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained Policy Optimization," *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," *arXiv preprint arXiv:1606.06565*, 2016.

[3] L. Gao, J. Schulman, and J. Hilton, "Scaling Laws for Reward Model Overoptimization," *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[4] D. Hadfield-Menell, S. Milli, P. Abbeel, S. Russell, and A. Dragan, "Inverse Reward Design," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[5] G. Iyengar, "Robust Dynamic Programming," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 257–280, 2005.

[6] J. Karwowski, O. Hayman, X. Bai, K. Kiendlhofer, C. Griffin, and J. Skalse, "Goodhart's Law in Reinforcement Learning," *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

[7] V. Krakovna et al., "Specification Gaming: The Flip Side of AI Ingenuity," *DeepMind Blog*, 2020. `https://deepmind.google/discover/blog/specification-gaming-the-flip-side-of-ai-ingenuity`

[8] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] T. Kwa, D. Thomas, and A. Garriga-Alonso, "Catastrophic Goodhart: Regularizing RLHF with KL Divergence Does Not Mitigate Heavy-Tailed Reward Misspecification," *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[10] R. Laroche, P. Trichelair, and R. Tachet des Combes, "Safe Policy Improvement with Baseline Bootstrapping," *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[11] J. Leike et al., "AI Safety Gridworlds," *arXiv preprint arXiv:1711.09883*, 2018.

[12] K. Nadjahi, R. Laroche, and R. Tachet des Combes, "Safe Policy Improvement with Soft Baseline Bootstrapping," *Proceedings of ECML-PKDD*, 2019.

[13] A. Nilim and L. El Ghaoui, "Robust Control of Markov Decision Processes with Uncertain Transition Matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[14] A. Pan, K. Bhatia, and J. Steinhardt, "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models," *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[16] S. Russell, *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking, 2019.

[17] J. Skalse, L. Farnik, S. R. Motwani, E. Jenner, A. Gleave, and A. Abate, "STARC: A General Framework for Quantifying Differences Between Reward Functions," *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[18] J. Skalse and A. Abate, "Quantifying the Sensitivity of Inverse Reinforcement Learning to Misspecification," *arXiv preprint arXiv:2212.07620*, 2024.

[19] Y. Zhang, Q. Vuong, and K. Ross, "FOCOPS: First Order Constrained Optimization in Policy Space," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

A Proof of Theorem 1

We prove both directions of the biconditional in (7).

Setup. Since $\pi \in \Pi_K$, equation (5) gives $\mu_\pi(s,a) = \mu_{\pi_0}(s,a)$ for all $(s,a) \notin K$, so $\Delta\mu(s,a) = 0$ for all $(s,a) \notin K$. Using the return representation (2):

$$\begin{aligned} J_R(\pi) - J_R(\pi_0) &= \sum_{s,a} \Delta\mu(s,a) \sum_{s'} T(s'|s,a)\, R(s,a,s') \\ &= \sum_{(s,a)\in K} \Delta\mu(s,a) \sum_{s'} T(s'|s,a)\, R(s,a,s') \end{aligned} \tag{11}$$

Adversarial minimisation. We compute $\min_{R\in\mathcal{U}_K}\big[J_R(\pi) - J_R(\pi_0)\big]$. The minimisation factorises over state-action pairs in K because each $R(s,a,s')$ appears in (11) only through its contribution to $\Delta\mu(s,a)\sum_{s'} T(s'|s,a)R(s,a,s')$, and the adversary can choose $R(s,a,s')$ independently within $[R'(s,a,s') - \varepsilon_{\text{in}},\ R'(s,a,s') + \varepsilon_{\text{in}}]$ for each (s,a,s').

Fix $(s,a) \in K$. The adversary minimises $\Delta\mu(s,a)\sum_{s'} T(s'|s,a)\, R(s,a,s')$ over $R(s,a,\cdot)$. Since $T(s'|s,a) \geq 0$ and $\sum_{s'} T(s'|s,a) = 1$:

- If $\Delta\mu(s,a) > 0$: set $R(s,a,s') = R'(s,a,s') - \varepsilon_{\text{in}}$ for all s'. The contribution becomes $\Delta\mu(s,a)\big[R'_K(s,a) - \varepsilon_{\text{in}}\big] = \Delta\mu(s,a)\, R'_K(s,a) - |\Delta\mu(s,a)|\, \varepsilon_{\text{in}}$.
- If $\Delta\mu(s,a) < 0$: set $R(s,a,s') = R'(s,a,s') + \varepsilon_{\text{in}}$ for all s'. The contribution becomes $\Delta\mu(s,a)\big[R'_K(s,a) + \varepsilon_{\text{in}}\big] = \Delta\mu(s,a)\, R'_K(s,a) - |\Delta\mu(s,a)|\, \varepsilon_{\text{in}}$.

- If $\Delta\mu(s,a) = 0$: any choice is optimal; contribution is 0.

In all cases the minimised contribution equals $\Delta\mu(s,a)\, R'_K(s,a) - |\Delta\mu(s,a)|\, \varepsilon_{\text{in}}$.

Closed form. Summing over all $(s,a) \in K$:

$$\min_{R \in \mathcal{U}_K} \big(J_R(\pi) - J_R(\pi_0)\big) = \sum_{(s,a)\in K} \Delta\mu(s,a)\, R'_K(s,a) - \varepsilon_{\text{in}} \sum_{(s,a)\in K} |\Delta\mu(s,a)| = \Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}} \|\Delta\mu_K\|_1 \tag{12}$$

Biconditional. Equation (12) shows that $\min_{R\in\mathcal{U}_K}(J_R(\pi) - J_R(\pi_0)) \geq 0$ if and only if $\Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}}\|\Delta\mu_K\|_1 \geq 0$, which completes the proof. □

Adversary's choice is optimal. The adversary's strategy (set R to the sign-opposite boundary pointwise) is both feasible (lies in $\mathcal{U}_K$) and attains the minimum. It is unique whenever $\Delta\mu(s,a) \neq 0$ for all $(s,a) \in K$.

B Lagrangian Derivation Details

B.1 Zero Duality Gap

The optimisation $\max_{\pi\in\Pi_K} J_{R'}(\pi)$ subject to $\Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}}\|\Delta\mu_K\|_1 \geq 0$ is a linear programme in the occupancy measure μ_π. For finite MDPs, the set of achievable occupancy measures is a polytope [15], and the constraint is affine in μ_π. Since the primal is a linear programme, strong duality holds unconditionally: $P^* = D^*$ without requiring Slater's condition.

B.2 Minimax Exchange

Substituting the ℓ_1–ℓ_∞ identity (9) into the Lagrangian (8) and collecting terms:

$$\begin{aligned} d(\lambda) &= \max_{\pi\in\Pi_K} J_{R'}(\pi) + \lambda\left[\Delta\mu_K \cdot R'_K - \varepsilon_{\text{in}} \max_{\|v\|_\infty \leq 1} v \cdot \Delta\mu_K\right] \\ &= \max_{\pi\in\Pi_K} \min_{\|v\|_\infty\leq 1} J_{R'}(\pi) + \lambda\, \Delta\mu_K \cdot (R'_K - \varepsilon_{\text{in}} v) \end{aligned} \tag{13}$$

The exchange $\max_\pi \min_v = \min_v \max_\pi$ holds by the minimax theorem (von Neumann, 1928) since: (i) Π_K is a compact convex polytope; (ii) $\{v : \|v\|_\infty \leq 1\}$ is compact and convex; (iii) the objective in (13) is linear (hence concave-convex) in (μ_π, v) jointly. After the exchange, the inner $\max_\pi$ for fixed v is:

$$\max_{\pi\in\Pi_K} \sum_{s,a} \mu_\pi(s,a)\big[(1+\lambda)R'(s,a) - \lambda\varepsilon_{\text{in}} v(s,a)\big] = \max_{\pi\in\Pi_K} J_{\tilde{R}^\lambda_v}(\pi)$$

which is standard policy iteration under the modified reward (10).

B.3 Optimality of $v^*(s,a) = \operatorname{sgn}(\Delta\mu(s,a))$

In the outer $\min_v \max_\pi$ problem, v^* maximises $-v \cdot \Delta\mu_K$ subject to $\|v\|_\infty \leq 1$. By the definition of the dual norm, the optimum is $v^*(s,a) = -\operatorname{sgn}(\Delta\mu(s,a))$ for the $\min_v$ direction, which after sign reversal gives $v^*(s,a) = \operatorname{sgn}(\Delta\mu(s,a))$ as stated in Section 5.

B.4 Practical Convergence

The alternating optimisation (update π, then $v = \text{sgn}(\Delta\mu)$, repeat) is not guaranteed to converge to a fixed point. The sign function is discontinuous at $\Delta\mu(s, a) = 0$, causing oscillation when occupancy differences are near zero. In practice we terminate on policy convergence rather than sign convergence, which is equivalent to solving a slightly perturbed problem. For the experiments reported here all runs converged within 20 inner iterations. A full convergence analysis is left to future work.

C Additional Experiments

Additional experimental results (hyperparameter sweeps, scaling experiments, stochastic transitions, and distractor location analysis) are available in the project repository and will be included in the full paper version.