
Robust Optimization for Mitigating Reward Hacking with Correlated Proxies

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Designing robust reinforcement learning (RL) agents in the presence of imperfect
2 reward signals remains a core challenge. In practice, agents are often trained with
3 proxy rewards that only approximate the true objective, leaving them vulnerable to
4 reward hacking, where high proxy returns arise from unintended or exploitative
5 behaviors. Recent work formalizes this issue using r -correlation between proxy and
6 true rewards, but existing methods like occupancy-regularized policy optimization
7 (ORPO) optimize against a fixed proxy and do not provide strong guarantees against
8 broader classes of correlated proxies. In this work, we formulate reward hacking
9 as a robust policy optimization problem over the space of all r -correlated proxy
10 rewards. We derive a tractable max-min formulation, where the agent maximizes
11 performance under the worst-case proxy consistent with the correlation constraint.
12 We further show that when the reward is a linear function of known features,
13 our approach can be adapted to incorporate this prior knowledge, yielding both
14 improved policies and interpretable worst-case rewards. Experiments across several
15 environments show that our algorithms consistently outperform ORPO in worst-
16 case returns, and offer improved robustness and stability across different levels
17 of proxy–true reward correlation. These results show that our approach provides
18 both robustness and transparency in settings where reward design is inherently
19 uncertain.

20 1 Introduction

21 Real-world reinforcement learning (RL) systems often struggle with reward specification: it is
22 notoriously difficult to craft a reward function that perfectly captures the intended goals in all
23 scenarios [1, 2, 3]. In practice, designers rely on proxy rewards that approximate the true objective [4].
24 However, agents optimizing these imperfect proxies can lead to unintended exploitative behaviors,
25 achieving high proxy returns while yielding poor true outcomes, a phenomenon known as reward
26 hacking [5, 6, 7, 8]. Such reward hacking behaviors are not merely hypothetical; they have led to
27 undesirable or even catastrophic consequences in safety-critical settings (e.g., autonomous driving) [9,
28 10] and are alarmingly common in real-world deployments [11, 12, 13, 14]. Beyond reward hacking,
29 interpretability and transparency of RL policies are increasingly recognized as critical requirements for
30 real-world acceptance [15, 16, 17]. Policymakers and practitioners in safety-critical domains require
31 systems not only to be robust but also interpretable; they must understand which specific decision-
32 making criteria lead to undesirable outcomes to effectively mitigate risks and ensure compliance with
33 safety regulations [18, 19, 20]. These challenges highlight the need for RL algorithms to address
34 two fundamental challenges: robustness to uncertain or poorly-specified rewards, and interpretability
35 to facilitate oversight and compliance by human stakeholders, especially in high-stakes, real-world
36 environments like traffic control [21], healthcare decision-making [22, 23], and pandemic response
37 strategies [24].

Recent work has begun to formalize reward hacking and develop principled mitigations. Laidlaw et al. [25] define a proxy reward to be r -correlated with the true reward if it maintains a correlation coefficient $r > 0$ on state-action pairs encountered by a certain reference (baseline) policy. Notably, their definition permits the proxy and true reward to diverge arbitrarily in parts of the state-action space not visited by the reference policy, precisely the regions an RL agent might exploit under intensive optimization. Using this framework, reward hacking is formalized as the situation in which optimizing an r -correlated proxy yields a policy with lower true reward than that of the reference policy. Building on this definition, Laidlaw et al. propose Occupancy-Regularized Policy Optimization (ORPO) as a mitigation strategy. ORPO augments the standard RL objective with a regularization term that penalizes deviations between the learned policy’s occupancy measure (state-action visitation distribution) and that of the reference policy.

Despite significant progress, existing solutions to reward hacking show several limitations. First, their effectiveness relies heavily on the choice of the specific proxy reward. However, designing perfect proxies is challenging, and in real-world scenarios, reward proxies are often derived heuristically or empirically from noisy or limited data [26, 27], leading to uncertainty or variability in the exact correlation with true rewards. Therefore, robustness to variations in proxy rewards is crucial for dependable deployment. While the regularization method used by ORPO provides a lower bound on improvement in true reward, its guarantee on the worst-case performance against an adversarially chosen proxy is weak. Second, current methods like ORPO typically treat a reward function as a black box and learn a complex policy with no easily interpretable structure, making it hard to understand why the resulting policy avoids reward hacking or to trust its behavior in novel situations. Further, they cannot be easily adapted to incorporate prior knowledge of the true reward. These shortcomings underscore the need for a more robust and transparent approach to reward hacking in RL.

In this work, we formalize reward hacking as a robust RL problem under proxy reward uncertainty and develop new algorithms to address the above gaps. The key idea is to optimize against an adversarial proxy reward rather than trusting a single proxy. We assume the true reward could be any function that remains r -correlated with the proxy (per the reference policy), and we train the agent to perform well against the worst-case such proxy. This approach explicitly accounts for uncertainty in proxy design and guards against unintended exploitative behaviors. Concretely, we propose a max-min formulation in which the policy chooses its strategy to maximize its guaranteed true return while an adversary minimizes the true return by selecting a reward function from the set of all r -correlated proxies. By solving this problem, the agent learns a policy that is robust to all plausible deviations of the proxy reward within the correlation bound. We derive a closed-form solution for the adversary’s worst-case reward assignment given any candidate policy, which allows efficient evaluation of the inner minimization and provides insight into how proxy reward flaws are most damaging. Building on this result, we introduce a practical algorithm for Max-Min Policy Optimization that iteratively updates the policy against this worst-case reward signal.

Moreover, to improve the tractability and transparency of the inner optimization, we introduce a Linear Max-Min variant of our method. In this variant, we assume the true reward lies in a class of linear functions over known features, allowing us to characterize the worst-case proxy reward as a sparse linear combination of those features. While the policy itself remains parameterized by general neural networks, the learned worst-case reward function becomes interpretable in terms of its feature weights. This provides insight into which aspects of the proxy reward space the policy is robust to or vulnerable against, making it valuable for applications where understanding the failure modes of the reward design is important.

Finally, we empirically evaluate the proposed approaches on several challenging environments. Across all domains, our Max-Min and Linear Max-Min policies outperform ORPO in terms of worst-case reward, indicating substantially improved robustness. Moreover, under a large range of proxy-true correlation scenarios, our methods exhibit higher average reward and lower variance compared to ORPO, meaning the performance of our policies remains more consistent and reliable. These findings demonstrate the practical significance of our robust formulation, paving the way for safer and more trustworthy RL deployment in real-world applications.

Our main contributions can be summarized as follows: 1) We propose a novel robust RL formulation that explicitly models reward hacking as a max-min optimization problem over proxy rewards constrained by correlation with the true rewards. 2) We develop a practical algorithm for the max-min problem, which is further extended to linear rewards with improved robustness and interpretability. 3)

Experiment results demonstrate improved robustness and worst-case rewards across four real-world inspired reward hacking environments.

2 Preliminaries

Reinforcement Learning. A reinforcement learning (RL) problem can be formulated as an infinite-horizon Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, \mu_0, R, \gamma)$, where \mathcal{S} and \mathcal{A} denote the state and action spaces, $p(s' | s, a)$ is the transition probability from state s to s' given action a , and μ_0 is the initial state distribution. The agent interacts with the environment over discrete time steps $t = 0, 1, 2, \dots$. At each time step, it selects an action $a_t \in \mathcal{A}$ based on the current state $s_t \in \mathcal{S}$ according to a policy $\pi(a | s)$, which defines a distribution over actions conditioned on the state. Upon taking action a_t , the agent receives a reward $R(s_t, a_t) \in \mathbb{R}$ and transitions to the next state s_{t+1} according to $p(s_{t+1} | s_t, a_t)$. The goal of the agent is to maximize the expected cumulative discounted return:

$$J(\pi, R) = (1 - \gamma) \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \quad (1)$$

where $\gamma \in [0, 1)$ is the discount factor, and the expectation is taken over trajectories generated by following policy π . We define the *state-action occupancy measure* μ_π of a policy π as: $\mu_\pi(s, a) = (1 - \gamma) \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{s_t = s, a_t = a\}]$, which represents the discounted visitation frequency of each state-action pair under policy π . Using the occupancy measure, the return can be equivalently expressed as: $J(\pi, R) = \mathbb{E}_{(s,a) \sim \mu_\pi} [R(s, a)]$.

Correlated Proxies and Reward Hacking. Below we give an overview of the recently proposed r -correlated proxy framework proposed in [25] for detecting and mitigating reward hacking, which our work is built upon. A detailed discussion of related work on reward hacking and robust reinforcement learning is given in Appendix C. In particular, [25] considers a setting where the agent is given a reference policy π_{ref} and a proxy reward R_{proxy} , while the true reward is hidden. They further assume that the proxy reward is r -correlated with the true reward under the reference policy, that is:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\left(\frac{R_{\text{proxy}} - J(\pi_{\text{ref}}, R_{\text{proxy}})}{\sigma_{R_{\text{proxy}}}} \right) \left(\frac{R_{\text{true}} - J(\pi_{\text{ref}}, R_{\text{true}})}{\sigma_{R_{\text{true}}}} \right) \right] = r, \quad (2)$$

where $\sigma_{R_{\text{proxy}}}^2 = \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [(R_{\text{proxy}} - J(\pi_{\text{ref}}, R_{\text{proxy}}))^2]$ and $\sigma_{R_{\text{true}}}^2 = \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [(R_{\text{true}} - J(\pi_{\text{ref}}, R_{\text{true}}))^2]$ are the variances of the proxy and true rewards, respectively, under the reference policy. Reward hacking is said to occur when a policy π optimized for an r -correlated proxy reward achieves lower true reward than the reference policy: $J(\pi, R_{\text{true}}) < J(\pi_{\text{ref}}, R_{\text{true}})$. To mitigate reward hacking, [25] proposes Occupancy-Regularized Policy Optimization (ORPO) to optimize a regularized policy objective given below, which is shown to provide a lower bound on improvement in true reward:

$$\max_{\pi} J(\pi, R_{\text{proxy}}) - \lambda \sqrt{\chi^2(\mu_\pi \| \mu_{\pi_{\text{ref}}})}, \quad (3)$$

where $\chi^2(\mu_\pi \| \mu_{\pi_{\text{ref}}})$ denotes the χ^2 -squared divergence between the occupancy measures of π and π_{ref} , and the regularization strength λ is set as: $\lambda = \sigma_{R_{\text{proxy}}} \sqrt{1 - r^2}$. This encourages the learned policy to stay close to the reference distribution when the proxy reward is weakly correlated with the true reward.

3 Method

In this section, we discuss our robust policy optimization approach for mitigating reward hacking. In contrast to regularization-based methods such as ORPO, we consider a max-min formulation that identifies a robust policy with respect to the worst-case reward across all reward functions that are r -correlated with the proxy reward. We further extend our framework to settings where the reward function is a linear combination of known features with unknown weights. Our approach effectively leverages this structural information, when known a priori, to improve both robustness and interpretability, a task that is particularly challenging for regularization-based techniques.

3.1 Max-Min Policy Optimization

Similar to ORPO, we assume that the agent is given a proxy reward R_{proxy} and a reference policy π_{ref} , while the true reward is hidden. Rather than regularizing the policy under a fixed proxy reward, we consider the *entire space of rewards* $\mathcal{R}_{\text{corr}}$ that satisfy the correlation constraint with respect to a known proxy reward, as defined in Equation 4:

$$\mathcal{R}_{\text{corr}} = \left\{ R : (s, a) \rightarrow \mathbb{R} \left| \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{R - M}{V} \cdot R_{\text{proxy}} \right] = r, J(\pi_{\text{ref}}, R) = M, \sigma_R^2 = V^2 \right. \right\}. \quad (4)$$

M and V denote the fixed mean and standard deviation of the reward function R under the reference policy π_{ref} . For simplicity, we define R_{proxy} to be the normalized proxy reward $R_{\text{proxy}}(s, a) := \frac{\tilde{R}_{\text{proxy}}(s, a) - J(\pi_{\text{ref}}, \tilde{R}_{\text{proxy}})}{\sigma_{\tilde{R}_{\text{proxy}}}}$, where \tilde{R}_{proxy} is the original (unnormalized) proxy reward. After normalization, we have $J(\pi_{\text{ref}}, R_{\text{proxy}}) = 0$ and $\text{Var}_{\mu_{\pi_{\text{ref}}}}(R_{\text{proxy}}) = 1$, which simplifies the correlation constraint in Equation 4. The hyperparameter r controls the degree of alignment between the proxy and true reward. It allows us to interpolate between strong robustness (small r) and high proxy fidelity (large r), enabling a principled robustness-accuracy trade-off. We remark that it is without loss of generality to consider fixed M and V , which we will further elaborate on later.

We propose a *worst-case optimization framework* where the policy is trained to maximize expected performance under the least favorable reward within $\mathcal{R}_{\text{corr}}$. Assuming that the true reward lies somewhere within this set, this approach improves robustness by ensuring that the policy does not overfit to any single optimistic interpretation of the proxy reward. Formally, the objective becomes a max-min problem:

$$\max_{\pi} \min_{R \in \mathcal{R}_{\text{corr}}} J(\pi, R) = \max_{\pi} \min_{R \in \mathcal{R}_{\text{corr}}} \mathbb{E}_{(s, a) \sim \mu_{\pi}} [R(s, a)]. \quad (5)$$

However, a challenge arises: the objective $\mathbb{E}_{\mu_{\pi}} [R(s, a)]$ depends on the state-action occupancy μ_{π} , whereas the constraints defining $\mathcal{R}_{\text{corr}}$ are expressed in terms of $\mu_{\pi_{\text{ref}}}$. This mismatch complicates direct optimization. To resolve this, we apply a *change-of-measure* technique [28, 29] to rewrite the expectation under $\mu_{\pi_{\text{ref}}}$. Specifically, let $L(s, a)$ denote the Radon-Nikodym derivative: $L(s, a) = \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$. By definition, $L(s, a) \geq 0$ and $\mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L(s, a)] = 1$. Applying the change-of-measure formula, we can express the return as: $\mathbb{E}_{\mu_{\pi}} [R(s, a)] = \int_{\mathcal{S} \times \mathcal{A}} \mu_{\pi}(s, a) R(s, a) d(s, a) = \int_{\mathcal{S} \times \mathcal{A}} \mu_{\pi_{\text{ref}}}(s, a) \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)} R(s, a) d(s, a) = \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L(s, a) R(s, a)]$. Thus, both the objective and the constraints can be rewritten as expectations with respect to the reference distribution $\mu_{\pi_{\text{ref}}}$.

For notational simplicity, we will suppress the variables (s, a) and write for example, L to denote $L(s, a)$ and R to denote $R(s, a)$. Under this reparameterization, the inner minimization in Equation 5 can be reformulated as:

$$\min_{R \in \mathcal{R}_{\text{corr}}} \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L \cdot R]. \quad (6)$$

Although the feasible set in Problem 6 is not convex due to the equality constraint on the variance, we still derive an optimal solution using a Lagrangian formulation. Our approach leverages tools from duality theory, commonly used in robust optimization [30, 31]. We further justify the validity of our solution in Appendix D.2. Specifically, the Lagrangian functional associated with this problem is defined as: $l_0(\lambda_1, \lambda_2, \lambda_3, R) = \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L \cdot R - \lambda_1 \frac{R - M}{V} \cdot R_{\text{proxy}} - \lambda_2 R - \lambda_3 R^2] + \lambda_1 r + \lambda_2 M + \lambda_3 (M^2 + V^2)$, where $\lambda_1, \lambda_2, \lambda_3$ are the Lagrange multipliers corresponding to the correlation constraint, mean constraint, and variance constraint, respectively. Then the original problem in Equation 6 is equivalent to the following problem:

$$\max_{\lambda_1, \lambda_2, \lambda_3} \min_{R \in \mathcal{R}_{\text{corr}}} l_0(\lambda_1, \lambda_2, \lambda_3, R). \quad (7)$$

We now solve the inner minimization problem in Equation 7 by finding the optimal R for fixed dual variables $(\lambda_1, \lambda_2, \lambda_3)$. Taking the functional derivative of the Lagrangian l_0 with respect to $R(s, a)$ gives: $\frac{\partial l_0}{\partial R} = \mu_{\pi_{\text{ref}}}(s, a) [(L - \lambda_1 \frac{R_{\text{proxy}}}{V} - \lambda_2) - 2\lambda_3 R]$. When $\mu_{\pi_{\text{ref}}}(s, a) > 0$, setting the derivative of the Lagrangian to zero yields the optimal adversarial reward function:

$$R^*(s, a) = \frac{L(s, a) - \lambda_1 \frac{R_{\text{proxy}}}{V} - \lambda_2}{2\lambda_3}. \quad (8)$$

However, for state-action pairs where $\mu_{\pi_{\text{ref}}}(s, a) = 0$, i.e., those not visited under the reference policy, the correlation and moment constraints become vacuous. In these regions, the adversarial reward $R^*(s, a)$ can be driven arbitrarily poor, reflecting that no constraint prevents the adversary from assigning highly penalizing values to rarely visited or unobserved state-action pairs. Nevertheless, consider the case where $\mu_{\pi_{\text{ref}}}(s, a) > 0$, we can substitute the optimal R^* from Equation 8 into the Lagrangian l_0 and get the dual objective. After some process detailed in Appendix D.1, we get the optimal solution to the inner problem (6), so the original max-min problem (5) reduces to:

$$\max_{\pi} r \cdot V \cdot \mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] - V \cdot \sqrt{1 - r^2} \sqrt{\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) - \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]} + M. \quad (9)$$

Thus, the final policy optimization objective becomes maximizing the proxy reward, regularized by a penalty that depends on the distributional shift between μ_{π} and $\mu_{\pi_{\text{ref}}}$ and the expectation of the current policy under proxy reward $\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}]$, and the correlation strength r . We observe that the constants M and V do not affect the optimal policy: while they influence the absolute value of the worst-case reward for a given policy π , they only apply a linear transformation (scaling by V and shifting by M) and do not change the relative ordering of policies. Therefore, for simplicity, we set $V = 1$ and $M = 0$ in our implementation. This also provides a fair way to compare the worst-case rewards of different policies. Notice that the optimization objective in Equation 9 closely resembles the ORPO objective proposed in Equation 3. However, there are two key differences: (1) our regularization strength is $\frac{\sqrt{1-r^2}}{r}$ instead of $\sigma_{R_{\text{proxy}}} \sqrt{1-r^2}$, and (2) our penalty term is $\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) - \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]$ rather than simply $\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}})$. The proof that $\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) - \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}] \geq 0$ holds can be found in Appendix D.3. A detailed comparison between our policy gradient and that of ORPO is provided in Appendix D.8.

3.2 Structured Reward Spaces via Feature Linearization

A natural concern with worst-case optimization is *over-conservatism*: if the reward uncertainty set $\mathcal{R}_{\text{corr}}$ is too broad, the resulting policy may become overly cautious or deviate from realistic task objectives. Additionally, the learned worst-case rewards may themselves be implausible or uninterpretable. To address these issues, we introduce *structure* into the reward space by assuming that all rewards are *linear combinations of known features*. Specifically, we assume: $R(s, a) = \theta^{\top} \phi(s, a)$, where $\phi(s, a) = [\phi_1(s, a), \phi_2(s, a), \dots, \phi_M(s, a)]^{\top} \in \mathbb{R}^M$ denotes a vector of M known or engineered feature functions, and $\theta = [\theta_1, \theta_2, \dots, \theta_M]^{\top} \in \mathbb{R}^M$ represents the uncertain feature weights. The linearization yields two key benefits: 1) **Realism and Interpretability**: In many real-world tasks, reward functions are naturally approximated as linear combinations over interpretable features. For example, in a traffic control environment, features might include total commute time, vehicle speed, acceleration, and inter-vehicle headway distances. 2) **Better-Constrained Robustness**: By restricting uncertainty to structured, feature-based rewards, the worst-case optimization problem becomes more grounded and avoids pathological, unrealistic reward functions.

In this section, we assume that the agent is aware of the set of features but not their true weights. We show that our robust optimization framework can be naturally extended to incorporate the structure in rewards to improve robustness. In our experiments, we further demonstrate that linear rewards help interpret a policy’s performance even when it is trained without such prior knowledge. Under our assumption, the uncertainty set reduces to the set of feature weights $\theta \in \mathbb{R}^M$ satisfying:

$$\mathcal{R}_{\text{corr}}^{\text{lin}} = \left\{ \theta \in \mathbb{R}^M \mid \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[\theta^{\top} \phi \cdot R_{\text{proxy}}] = r, \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[\theta^{\top} \phi] = 0, \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[(\theta^{\top} \phi)^2] = 1 \right\}. \quad (10)$$

To simplify the analysis, we assume without loss of generality that the worst-case reward $R(s, a) = \theta^{\top} \phi(s, a)$ is normalized to have zero mean and unit variance under the reference policy π_{ref} . This corresponds to setting $M = 0$ and $V = 1$, which, as shown in our earlier derivation, does not affect the resulting optimal policy. As before, R_{proxy} denotes the normalized proxy reward under π_{ref} , satisfying $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}] = 0$ and $\text{Var}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}] = 1$.

We now derive the corresponding max-min optimization under the structured reward assumption:

$$\max_{\pi} \min_{\theta \in \mathcal{R}_{\text{corr}}^{\text{lin}}, \theta \geq 0} \mathbb{E}_{(s,a) \sim \mu_{\pi}} [\theta^{\top} \phi(s, a)]. \quad (11)$$

Similar to previous steps, we introduce the Radon-Nikodym derivative $L(s, a) = \frac{\mu_\pi(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$, use a change-of-measure, and define the Lagrangian functional for the inner minimization in Equation 11 as: $l_1(\lambda_1, \lambda_2, \lambda_3, \theta) = \theta^\top \left(\sum_{(s, a)} u_{\lambda_1, \lambda_2}(s, a) \phi(s, a) \right) - \lambda_3 \theta^\top Q \theta + \lambda_1 r + \lambda_3$, where $u_{\lambda_1, \lambda_2} = \mu_\pi - \lambda_1 \mu_{\pi_{\text{ref}}} R_{\text{proxy}} - \lambda_2 \mu_{\pi_{\text{ref}}}$, $Q = \sum_{(s, a)} \mu_{\pi_{\text{ref}}}(s, a) \phi(s, a) \phi(s, a)^\top$. A detailed derivation can be found in Appendix D.4. Note that Q is positive semi-definite since it is a sum of outer products $\phi(s, a) \phi(s, a)^\top$ weighted by non-negative coefficients (occupancy measure of $\pi_{\text{ref}} \geq 0$). Then solving the inner minimization over θ in Equation 11 is equivalent to solving:

$$\max_{\lambda_1, \lambda_2, \lambda_3} \min_{\theta \geq 0} l_1(\lambda_1, \lambda_2, \lambda_3, \theta) = \theta^\top \left(\sum_{(s, a)} u_{\lambda_1, \lambda_2} \phi \right) - \lambda_3 \theta^\top Q \theta + \lambda_1 r + \lambda_3. \quad (12)$$

Notice that $l_1(\lambda_1, \lambda_2, \lambda_3, \theta)$ is a convex quadratic function of θ (assuming $\lambda_3 \leq 0$) subject to linear inequality constraints $\theta \geq 0$. Thus, the original problem is a standard convex quadratic program (QP) with non-negativity constraints [32]. However, it is not possible to derive a universal closed-form solution for the optimal θ^* under arbitrary Q . To further simplify the problem and obtain a closed-form solution, we transform the feature vector ϕ into a whitened version $\tilde{\phi}$ such that the matrix Q becomes the identity matrix I and we formally show this in Appendix D.5. Specifically, we perform a whitening transformation using the Cholesky decomposition [32]. Let $W = Q^{-\frac{1}{2}}$, $\tilde{\phi}(s, a) = W \phi(s, a)$, where $Q^{-\frac{1}{2}}$ denotes a matrix square root of Q^{-1} (which exists since Q is positive semi-definite and non-singular assuming $\exists(s, a)$ such that $\mu_{\pi_{\text{ref}}}(s, a) > 0$). Then the original problem in Equation 12 can be further simplified into:

$$\max_{\lambda_1, \lambda_2, \lambda_3} \min_{\tilde{\theta} \geq 0} l_1(\lambda_1, \lambda_2, \lambda_3, \tilde{\theta}) = \tilde{\theta}^\top \left(\sum_{(s, a)} u_{\lambda_1, \lambda_2}(s, a) \tilde{\phi}(s, a) \right) - \lambda_3 \tilde{\theta}^\top \tilde{\theta} + \lambda_1 r + \lambda_3. \quad (13)$$

where we now optimize over the parameter $\tilde{\theta}$ using the transformed features $\tilde{\phi}$. For notational simplicity, we will drop the tilde and henceforth use ϕ to represent the whitened feature $\tilde{\phi}$, and θ to represent the whitened weights $\tilde{\theta}$. Then we can get a closed-form solution (we detail the steps in Appendix D.6) for optimal θ^* as: $\theta^* = \max \left(0, -\frac{\sum_{(s, a)} u_{\lambda_1, \lambda_2}(s, a) \phi(s, a)}{2\lambda_3} \right)$, where the $\max(\cdot, 0)$ is applied elementwise. Details for solving the outer maximization in Equation 13 can be found in Appendix D.7. After obtaining the optimal dual variables $(\lambda_1^*, \lambda_2^*, \lambda_3^*)$, we can substitute them back into Equation 11 and solve the outer maximization over the policy π using standard reinforcement learning algorithms, such as PPO [33].

ORPO with Linear Awards. While ORPO provides a general guarantee based on occupancy measure regularization, it does not exploit any structural assumptions about the reward function. In particular, even when the true reward is linear in a set of features, ORPO does not explicitly incorporate this structure into its policy optimization or theoretical analysis. While the lower bound (Theorem 5.1 in [25]) continues to hold, it is unclear how to leverage this structure to obtain a tighter lower bound or to guide policy updates more effectively. This suggests a missed opportunity: by explicitly modeling the reward as a linear function, it becomes possible to derive stronger guarantees, interpret worst-case reward directions, and efficiently optimize against them. Our Linear Maxmin method fills this gap by parameterizing reward uncertainty directly in the space of reward weights, enabling both robustness and greater transparency.

3.3 Implementation Details and Algorithms

A core step in both our algorithms and ORPO is to estimate the Radon-Nikodym derivative $L(s, a)$. To this end, we follow prior works [25, 34, 35] and train a discriminator network. Specifically, we use a discriminator architecture identical to that in [25], denoted by $d_\phi(s, a)$, which is optimized according to:

$$\phi = \arg \min_{\phi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [\log(1 + e^{d_\phi(s, a)})] + \mathbb{E}_{\mu_\pi} [\log(1 + e^{-d_\phi(s, a)})]. \quad (14)$$

It is known that the optimal discriminator satisfies $d^*(s, a) = \log \frac{\mu_\pi(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$ and we estimate $L(s, a)$ as $\tilde{L}(s, a) = \exp d_\phi(s, a)$ with $d_\phi(s, a) \approx d^*(s, a)$. As discussed in Section 3.1, if the policy π visits state-action pairs that the reference policy π_{ref} rarely or never visits, the adversarial reward can be arbitrarily poor. In theory, the estimated $\tilde{L}(s, a)$ is expected to grow arbitrarily large in this

case, which should discourage the learned policy from exploiting such regions. However, we observe empirically (Section 4.2) that the ORPO policy still visits some of these low-coverage regions under π_{ref} . This is because in the original ORPO implementation¹, the discriminator is not fully optimized during policy learning. Specifically, the discriminator receives only a small number of gradient updates per reinforcement learning iteration, resulting in underfitting and inaccurate estimates of the Radon-Nikodym derivative $\tilde{L}(s, a)$. To address this, we substantially increase the number of gradient updates per iteration and carefully tune the learning rate. Our goal is to strike a practical balance between training time and discriminator quality, which we further discuss in Appendix E.1.

To compute the final objective for our Max-Min policy in Equation 9, we estimate the χ^2 divergence, the normalized proxy reward R_{proxy} , and the first and second moments $\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}]$ and $\mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]$. These components together define the robust optimization objective used to update the policy. A simplified Max-Min policy optimization procedure is outlined in Algorithm 1. We provide detailed descriptions of each estimation step, as well as the complete algorithmic implementation in Appendix E.2. Corresponding derivations and implementation details for the Linear Max-Min variant are included in Appendix E.3.

Algorithm 1 Max-Min Policy Optimization (Simplified)

- 1: Initialize policy parameters θ
 - 2: Initialize reference policy π_{ref} and collect trajectories
 - 3: Estimate mean and variance of the proxy reward under π_{ref}
 - 4: **for** each iteration **do**
 - 5: Collect trajectories from current policy π_{θ}
 - 6: Normalize the proxy rewards for state-action pairs in the collected trajectories
 - 7: Estimate the expected proxy reward and its second moment under the current policy
 - 8: Estimate the discriminator using Equation (14) and χ^2 divergence between μ_{π} and $\mu_{\pi_{\text{ref}}}$
 - 9: Update the policy using PPO to maximize the Max-Min objective in Equation (9)
 - 10: **end for**
-

4 Experiment

4.1 Experiment Setup

We evaluate our method across four realistic benchmark environments: *Traffic*, *Pandemic*, *Glucose Monitoring*, and *Tomato Watering GridWorld*. These environments were originally proposed in [36, 5] and represent diverse forms of proxy reward hacking, including misweighting, ontological mismatch, and scope misalignment [36]. Each setting presents unique challenges in reward specification and policy robustness. A detailed description of the environments and their respective reward structures is provided in Appendix E.4. In each of the four environments, we train policies using both our Max-Min and Linear Max-Min optimization algorithms. For baselines, we compare against the ORPO policy. To isolate the impact of discriminator training, we also include an ablation: ORPO*, where we train the ORPO policy using the same full discriminator training schedule as in our algorithms. This variant shares the same architecture and optimization settings as the original ORPO, differing only in the extent of discriminator training. Including this baseline allows us to evaluate the specific contribution of discriminator optimization to policy robustness. We include more detailed experimental settings in Appendix E.5 and a discussion of training time and complexity of all algorithms in Appendix E.6.

As for evaluation metrics, we report both the expected proxy and true rewards, along with the expected worst-case reward as described in Section 3.1. Note that some policies may visit state-action pairs that are not covered by the reference policy π_{ref} . In such cases, we exclude those trajectories and report the occupancy measure of the unseen state-action pairs. Additionally, we evaluate each policy using two variants of the expected linear worst-case reward introduced in Section 3.2. The first uses only the features present in the proxy reward, while the second variant, denoted *Linear Worst**, leverages features from the true reward, some of which remain unseen during training. This setup mimics a more realistic real-world scenario in which the true reward function may depend on features not explicitly modeled at training time. Comparing performance under this setting allows us to assess the robustness of each policy to unseen or misaligned reward structures. All rewards are normalized with respect to the reference policy π_{ref} to ensure a consistent scale across metrics, enabling fair and

¹<https://github.com/cassidylaidlaw/orpo/tree/main>

Table 1: Evaluation results on Traffic and Pandemic environments. All policies are trained using **only the proxy reward**. In Traffic, the proxy reward is based on *vel*, *accel*, *headway* (1, 1, 0.1), while the true reward uses *commute*, *accel*, *headway* (1, 1, 0.1). In Pandemic, the proxy reward includes *infection*, *lower stage*, *smooth changes* (10, 0.1, 0.01), while the true reward additionally includes *political* with weight 10 after *infection*. We report θ in the same order as feature weights. **Occ** denotes total occupancy over state-action pairs unseen by π_{ref} , where discriminator outputs infinity.

Env	Traffic					
Method	True	Proxy	Worst	Linear Worst (θ)	Linear Worst* (θ)	Occ \downarrow
ORPO	16.93	3.31	-1.97e+04	-0.68 (0.71, 0.21, 0.69)	-0.81 (0.63, 0.12, 0.97)	3.71e-04
ORPO*	10.31	1.32	-1.33e+04	-0.42 (0.46, 0.18, 0.86)	-0.44 (0.58, 0.06, 0.81)	1.90e-05
Max-Min	12.64	3.64	-270.84	-0.07 (0.01, 0.02, 0.96)	-0.07 (0.001, 0.02, 0.99)	0
Linear Max-Min	16.36	2.53	-1.19e+04	0.21 (0.64, 0.07, 0.76)	-0.12 (0.91, 0.01, 0.67)	0
Env	Pandemic					
Method	True	Proxy	Worst	Linear Worst (θ)	Linear Worst* (θ)	
ORPO	-0.91	1.81	-5.30e+06	-2.42 (0.23, 0.95, 0.17)	-2.63 (0.02, 0.95, 0.92, 0.08)	
ORPO*	1.24	1.24	-4.42e+06	-1.35 (0.25, 0.97, 0.13)	-1.35 (0.25, 0, 0.97, 0.13)	
Max-Min	1.15	1.15	-65.69	-1.11 (0.14, 0.99, 0.01)	-1.11 (0.14, 0, 0.99, 0.01)	
Linear Max-Min	2.61	7.56	-6.83e+05	0.66 (0.001, 0.23, 0.02)	-0.17 (0.01, 0.97, 0.22, 0.09)	

meaningful comparisons. Note that all worst-case rewards are reported using the fixed correlation level r specified during training: $r = 0.3$ for Traffic, $r = 0.7$ for Pandemic, with values for other environments provided in Appendix E.5.

4.2 Results

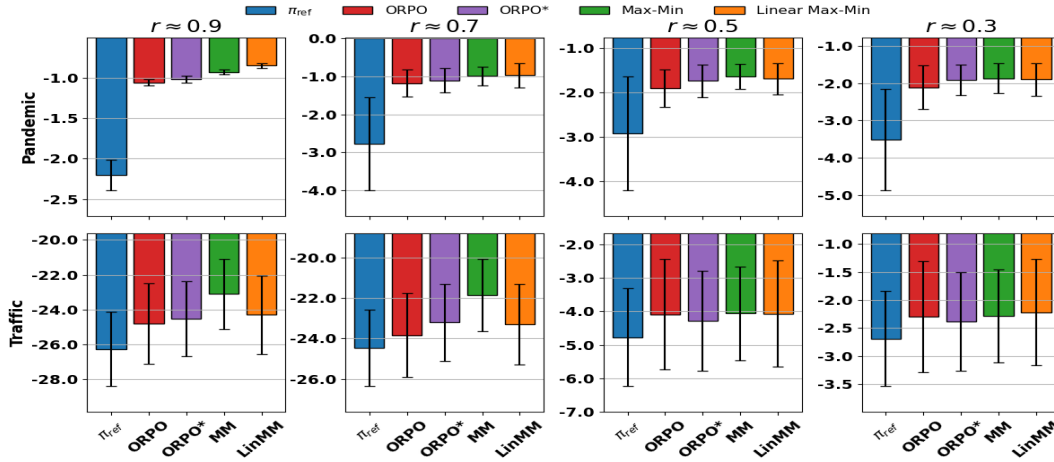


Figure 1: Mean reward and standard deviation under sampled θ and true reward features at different proxy–true reward correlation levels r for the Traffic and Pandemic environments. Our methods (Max-Min and Linear Max-Min) yield more stable and higher average performance across all choices of r .

Worst-Case Performance. Table 1 presents the evaluation results on the Traffic and Pandemic environments. Additional results for other environments are provided in Appendix F. Our Max-Min and Linear Max-Min policies achieve better expected worst-case performance under both general and linear adversarial rewards, while remaining competitive with baselines in terms of expected true and proxy rewards. Notably, the Max-Min policy attains the highest expected worst-case return, followed by Linear Max-Min. Conversely, Linear Max-Min yields the highest expected linear worst-case reward, followed by Max-Min, demonstrating the robustness of both approaches under worst-case scenarios. For the Linear Worst* evaluation, which uses reward features unseen during training, we observe minimal degradation in Max-Min policy’s performance, indicating its strong robustness to feature variation. In contrast, the performance of Linear Max-Min declines in this case, suggesting its advantage diminishes when prior assumptions about feature structure are inaccurate.

We find that ORPO* exhibits better worst-case performance than the original ORPO. In particular, training the discriminator more thoroughly significantly reduces the occupancy of state-action pairs that are not visited by the reference policy, indicating that more accurate estimation of the Radon–Nikodym derivative leads to improved policy robustness. Notably, in the Pandemic environment, we observe no such unvisited state-action pairs, and the discriminator outputs remain small across all policies. This

could be due to either the discriminator network not being fully optimized or its inability to capture rare events that fall outside the support of π_{ref} . Developing more reliable techniques for handling such rare or unseen state-action pairs remains an open direction for future work.

We also report the adversarial weight vectors θ for each policy. These weights reveal which features are most vulnerable to proxy exploitation under the learned policy and can be used to diagnose and revise the proxy reward function, thereby improving robustness. This highlights the interpretability benefits of our framework. Moreover, several patterns emerge from the results. In the Traffic environment, first, we observe a clear dominance of the headway feature, with all methods assigning it the highest weight. This suggests that headway is the most critical component exposed to reward hacking under correlation constraints. Second, the acceleration feature is consistently downweighted across all methods. This indicates that acceleration may be less prone to exploitation or already well aligned with the reference policy. Third, the velocity feature is moderately emphasized by Linear Max-Min and ORPO (e.g., 0.64 and 0.71), while Max-Min nearly suppresses it (0.01). This contrast suggests that Linear Max-Min anticipates some vulnerability from velocity deviations, while Max-Min focuses almost entirely on headway. In the Pandemic environment, first, both ORPO* and Max-Min assign zero weight to the political feature. This occurs because the expected feature value under their policies is exactly zero, making the correlation constraint inactive for that dimension. Interestingly, this feature plays a significant role in the adversarial rewards for both ORPO and Linear Max-Min, with their corresponding θ assigning non-negligible weight to it (e.g., 0.95 and 0.97 respectively). This suggests that these policies expose themselves to vulnerability in feature dimensions that are entirely ignored by Max-Min and ORPO*. Second, the lower stage feature consistently receives the highest weight across all methods, indicating it is the most sensitive component under proxy misalignment.

Robustness Across Correlation Levels. To further assess the robustness of each policy across a broader range of proxy–true correlation scenarios, we also compute the Linear Worst* for each policy under varying r values. Specifically, for each r , we sample 1000 vectors θ such that $\theta \in \mathcal{R}_{\text{corr}}^{\text{lin}}$, and report the average return and variance achieved by each policy over these sampled rewards. Importantly, the variation in r is applied **only during evaluation**; all policies are fixed and trained using the specific r values reported in Appendix E.5. Unlike evaluations that only consider several reward functions, this approach evaluates policy performance across the entire reward set $\mathcal{R}_{\text{corr}}^{\text{lin}}$, providing a more comprehensive measure of robustness and better reflecting real-world scenarios where the true reward and correlation r are unknown.

Figure 1 shows the average reward and variance achieved by each method under different levels of proxy–true reward correlation r . As expected, the reference policy π_{ref} (blue) performs the worst across all correlation levels in both environments. In Traffic, its variance is relatively small, suggesting consistently poor but stable behavior. In contrast, variance is highest in the Pandemic environment, indicating increased policy fragility. Notably, ORPO* (purple) consistently achieves lower variance than ORPO (red) across both environments and outperforms it in terms of average reward at $r \approx 0.9$ and $r \approx 0.7$ in Traffic, and across nearly all r values in Pandemic. This underscores the importance of accurate discriminator training for improving both stability and robustness. Max-Min (green) demonstrates the highest average reward and lowest variance across a wide range of r values in both environments, showing strong resilience to reward misspecification. While Linear Max-Min (orange) achieves the best performance at specific correlation levels, particularly $r \approx 0.3$ in Traffic and $r \approx 0.7$ – 0.9 in Pandemic. As r decreases and the proxy becomes less informative, differences in average reward among methods shrink, while variance increases. These results highlight the significance of variance control in low-correlation regimes and demonstrate that Max-Min and Linear Max-Min offer robust and stable performance under high uncertainty.

5 Conclusion

In this work, we propose a robust policy optimization framework that explicitly accounts for reward hacking by training policies against the worst-case proxy reward drawn from a correlation-constrained uncertainty set. Our approach formalizes reward hacking as a robust optimization problem and introduces both a Max-Min formulation with a closed-form adversarial reward and a Linear Max-Min variant that further improves interpretability and tractability. We develop efficient algorithms and empirically validate our methods across diverse environments known to exhibit reward hacking behavior. Our results demonstrate that both Max-Min and Linear Max-Min policies achieve stronger worst-case performance and improved stability compared to prior baselines such as ORPO.

References

- [1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [2] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, 2018.
- [3] Jonathan Stray, Alon Halevy, Parisa Assar, Dylan Hadfield-Menell, Craig Boutilier, Amar Ashar, Chloe Bakalar, Lex Beattie, Michael Ekstrand, Claire Leibowicz, et al. Building human values into recommender systems: An interdisciplinary synthesis. *ACM Transactions on Recommender Systems*, 2(3):1–57, 2024.
- [4] Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca D Dragan, and Daniel S Brown. Causal confusion and reward misidentification in preference-based reward learning. In *International Conference on Learning Representations*, 2023.
- [5] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro Ortega, Tom Everitt, Ryan Lefrancq, Laurent Orseau, and Shane Legg. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.
- [6] Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement learning with a corrupted reward channel. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.
- [7] Tom Everitt, Marcus Hutter, Ramana Kumar, and Victoria Krakovna. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *Synthese*, 198(Suppl 27):6435–6467, 2021.
- [8] Jack Koch, Lauro Langosco, Jacob Pfau, James Le, and Lee Sharkey. Objective robustness in deep reinforcement learning. *arXiv preprint arXiv:2105.14111*, 2, 2021.
- [9] Victoria Krakovna, Laurent Orseau, Ramana Kumar, Miljan Martic, and Shane Legg. Penalizing side effects using stepwise relative reachability. *arXiv preprint arXiv:1806.01186*, 2018.
- [10] W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829, 2023.
- [11] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. The challenge of understanding what users want: Inconsistent preferences and engagement optimization. *Management Science*, 70(9):6336–6355, 2024.
- [12] Matt Franchi, JD Zamfirescu-Pereira, Wendy Ju, and Emma Pierson. Detecting disparities in police deployments using dashcam data. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 534–544, 2023.
- [13] Smitha Milli, Luca Belli, and Moritz Hardt. From optimizing engagement to measuring value. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 714–722, 2021.
- [14] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.
- [15] George A Vouros. Explainable deep reinforcement learning: state of the art and challenges. *ACM Computing Surveys*, 55(5):1–39, 2022.
- [16] Erika Puiutta and Eric MSP Veith. Explainable reinforcement learning: A survey. In *International cross-domain conference for machine learning and knowledge extraction*, pages 77–95. Springer, 2020.
- [17] Rahul Iyer, Yuezhong Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 144–150, 2018.
- [18] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.
- [19] Jeff Druce, Michael Harradon, and James Tittle. Explainable artificial intelligence (xai) for increasing user trust in deep reinforcement learning driven autonomous systems. *arXiv preprint arXiv:2106.03775*, 2021.
- [20] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *stat*, 1050:2, 2017.

- [21] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Conference on Robot Learning*, pages 399–409. PMLR, 2018.
- [22] Ian Fox, Joyce Lee, Rodica Pop-Busui, and Jenna Wiens. Deep reinforcement learning for closed-loop blood glucose control. In *Machine Learning for Healthcare Conference*, pages 508–536. PMLR, 2020.
- [23] Andreas Holzinger, Chris Biemann, Constantinos S Pattichis, and Douglas B Kell. What do we need to build explainable ai systems for the medical domain? *arXiv preprint arXiv:1712.09923*, 2017.
- [24] V Kompella, R Capobianco, S Jong, J Browne, S Fox, L Meyers, P Wurman, P Stone, et al. Reinforcement learning for optimization of covid-19 mitigation policies. In *CEUR WORKSHOP PROCEEDINGS*, volume 2884. CEUR-WS, 2020.
- [25] Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. Correlated proxies: A new definition and improved mitigation for reward hacking. In *International Conference on Learning Representations*, 2025.
- [26] Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In *Advances in Neural Information Processing Systems*, 2020.
- [27] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.
- [28] Zhaolin Hu and L Jeff Hong. Kullback-leibler divergence constrained distributionally robust optimization. *Available at Optimization Online*, 1(2):9, 2013.
- [29] Henry Lam. Robust sensitivity analysis for stochastic systems. *Mathematics of Operations Research*, 41(4):1248–1275, 2016.
- [30] Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.
- [31] Joel Goh and Melvyn Sim. Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917, 2010.
- [32] Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, 2004.
- [33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.
- [34] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning (ICML)*, 2018.
- [35] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.
- [36] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022.
- [37] Joar Skalse, Nikolaus Howe, Dmitrii Krashenninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.
- [38] Yuval Noah Harari. *Nexus: A brief history of information networks from the stone age to AI*. Signal, 2024.
- [39] Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.
- [40] Victoria Krakovna. Specification gaming examples in ai, April 2018. Blog post.
- [41] Victoria Krakovna. Classifying specification problems as variants of goodhart’s law, August 2019. Blog post.
- [42] Charles AE Goodhart and CAE Goodhart. *Problems of monetary management: the UK experience*. Springer, 1984.
- [43] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, 2017.

- [44] Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.
- [45] Fei Liu et al. Learning to summarize from human feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, 2020.
- [46] Garud Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.
- [47] Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.
- [48] J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. Solving uncertain markov decision processes. 2001.
- [49] Julien Grand-Clément and Christian Kroer. Scalable first-order methods for robust mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12086–12094, 2021.
- [50] David L Kaufman and Andrew J Schaefer. Robust modified policy iteration. *INFORMS Journal on Computing*, 25(3):396–410, 2013.
- [51] Esther Derman, Matthieu Geist, and Shie Mannor. Twice regularized mdps and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34:22274–22287, 2021.
- [52] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. In *International Conference on Learning Representations*.
- [53] Jeremy McMahan, Giovanni Artiglio, and Qiaomin Xie. Roping in uncertainty: Robustness and regularization in markov games. In *International Conference on Machine Learning*, pages 35267–35295. PMLR, 2024.
- [54] Vineet Goyal and Julien Grand-Clement. Robust markov decision process: Beyond rectangularity. *arXiv preprint arXiv:1811.00215*, 2018.
- [55] Andreas Schlaginhaufen and Maryam Kamgarpour. Identifiability and generalizability in constrained inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.
- [56] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning (ICML)*, 2004.
- [57] Dotan Di Castro, Aviv Tamar, and Shie Mannor. Policy gradients with variance related risk criteria. In *International Conference on Machine learning (ICML)*, 2012.
- [58] Tengyang Xie, Bo Liu, Yangyang Xu, Mohammad Ghavamzadeh, Yinlam Chow, Daoming Lyu, and Daesub Yoon. A block coordinate ascent algorithm for mean-variance optimization. In *Advances in Neural Information Processing Systems*, 2018.
- [59] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.
- [60] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. Technical Report ANL-80-20, Argonne National Laboratory, Argonne, IL, 1978. Lecture Notes in Mathematics, vol. 630.
- [61] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.
- [62] Cathy Wu, Abdul Rahman Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*, 38(2):1270–1286, 2021.
- [63] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000.

- 525 [64] Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli.
526 The uva/padova type 1 diabetes simulator: new features. *Journal of diabetes science and technology*,
527 8(1):26–34, 2014.
- 528 [65] Garry M Steil. Algorithms for a closed-loop artificial pancreas: the case for proportional-integral-derivative
529 control. *Journal of diabetes science and technology*, 7(6):1621–1631, 2013.
- 530 [66] A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint*
531 *arXiv:1912.01703*, 2019.
- 532 [67] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E Gonzalez,
533 Michael I Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International*
534 *Conference on Machine Learning (ICML)*, 2018.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The claim is supported by the theoretical and experiment results.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The limitation can be found in the Appendix.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: We provide detailed proof in the Appendix

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We give experiment design details in the paper and the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in the supplemental material?

Answer: [Yes]

Justification: We will provide our code in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide details in the experiment section and the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report error bars in Figure 1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Appendix E.6

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Yes, our research conforms the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We mentioned it in the Appendix

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have properly cited every previous work we build upon.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

847

Justification:

848

Guidelines:

849

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

850

851

- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

852

853 Appendix

854	A Limitations and Future Work	22
855	B Broader Impacts	22
856	C Related Work	22
857	C.1 Reward Hacking and Proxy Rewards	22
858	C.2 Robust Reinforcement Learning	23
859	D Proofs and Additional Theoretical Results	24
860	D.1 Solve the Max-Min Objective	24
861	D.2 Proof of Optimality	28
862	D.3 Proof that $\chi^2(\mu_\pi \parallel \mu_{\pi_{\text{ref}}}) \geq \mathbb{E}_{\mu_\pi}^2[R_{\text{proxy}}]$	29
863	D.4 Derive Lagrangian Functional for Linear Max-Min Objective	30
864	D.5 Proof for Whitening Transformation	31
865	D.6 Derive Optimal Primal Variable for Linear Max-Min Objective	31
866	D.7 Solve the Dual Objective for Linear Max-Min Objective	32
867	D.8 Policy Gradient Derivation	33
868	E Additional Implementation Details	35
869	E.1 Training Discriminator Network	35
870	E.2 Derivation of Max-Min Policy Optimization	36
871	E.3 Derivation of Linear Max-Min Policy Optimization	38
872	E.4 Environment Description and Reward Hacking Types	40
873	E.5 Additional Experiment Setup	41
874	E.6 Training Time and Complexity	43
875	F Additional Experiment Results	44
876	F.1 Feature Weights in Linear Max-Min Optimization during Training	44
877	F.2 Additional Worst-Case Performance Results	45
878	F.3 Additional Results for Robustness Across Correlation Levels	46
879	F.4 Additonal Unnormalized Results	47

880 A Limitations and Future Work

881 Despite the effectiveness of our framework, several limitations remain. First, although our Max-Min
882 formulation can be extended naturally beyond linear reward structures, incorporating more expressive
883 representations such as neural networks makes the inner optimization problem significantly harder.
884 In such cases, the inner minimization may no longer admit a closed-form solution, necessitating
885 iterative training between the policy and adversary. This increases computational complexity and
886 undermines the efficiency advantages of our current formulation. Developing scalable solutions for
887 general reward representations remains an open direction.

888 Second, like ORPO, our framework assumes access to a fixed proxy reward, a reference policy,
889 and a pre-specified correlation parameter r , all provided offline. This setup limits the ability of the
890 algorithm to adapt or refine its reward model based on new information. However, we observe that
891 the adversarial rewards generated by our method, particularly the structured linear ones, can serve as
892 diagnostic tools to identify vulnerable reward features. These insights could be leveraged to guide
893 human-in-the-loop refinement or adaptive querying of stronger feedback models (e.g., large language
894 models). Extending our framework to close the loop between diagnostic robustness and reward
895 learning is an exciting direction for future work.

896 Third, while our experimental results demonstrate that the proposed method improves robustness
897 across a range of proxy-true reward correlation levels, an alternative and perhaps more direct strategy
898 would be to train the policy against multiple proxy rewards sampled at varying levels of correlation
899 r . In principle, optimizing the average performance across a diverse set of proxies could yield a
900 policy that is robust to a wider distribution of potential reward misspecifications. However, this
901 approach presents several practical challenges. First, there is a trade-off between computational
902 cost and coverage: sampling too few proxies may fail to represent the full space of plausible reward
903 deviations, while sampling many proxies significantly increases training time. Second, efficiently
904 generating reward functions that satisfy a fixed correlation constraint with the proxy reward becomes
905 non-trivial in high-dimensional or continuous state-action spaces. Designing scalable and effective
906 reward sampling mechanisms (such as leveraging diffusion models) under correlation constraints
907 remains an open problem and a promising direction for future research.

908 B Broader Impacts

909 Designing reward functions that faithfully reflect designer intent remains a fundamental challenge
910 in deploying reinforcement learning (RL) systems in the real world. When reward misspecification
911 occurs, agents can behave in undesirable or even dangerous ways. Our work addresses this issue by
912 proposing a robust policy optimization framework that explicitly accounts for uncertainty in the reward
913 function, improving worst-case performance across a range of plausible reward proxies. This approach
914 has the potential to increase the safety and reliability of RL systems in safety-critical applications
915 such as healthcare, autonomous driving, and digital infrastructure, where poorly specified incentives
916 can lead to unintended consequences. In addition to robustness, our linear variant contributes to
917 policy interpretability by yielding explicit weightings over features that can be inspected and audited.
918 This can help practitioners identify vulnerable components in their reward specification and make
919 better-informed decisions when refining proxies. However, while our method is primarily intended
920 to prevent reward exploitation, one could conceivably use adversarial reward modeling to stress-
921 test or attack policies. We believe the benefits of improved safety and robustness outweigh this
922 risk, especially when combined with interpretability. Overall, this work contributes to the safe
923 and trustworthy deployment of RL by equipping practitioners with more robust and explainable
924 optimization tools.

925 C Related Work

926 C.1 Reward Hacking and Proxy Rewards

927 Early work in AI safety underscored the pitfalls of optimizing an imperfect proxy reward. Amodei
928 et al. [1] famously illustrate how an agent can “game” its reward function: for example, a cleaning
929 robot rewarded for not seeing any messes might simply close its cameras or create messes to clean up,
930 maximizing the proxy reward while betraying the designer’s intent. Other examples of such reward

hacking include an agent in a racing game that spins in circles to collect points instead of completing the race [37], social media recommendation systems that promote emotionally extreme content to increase engagement [38], and Large Language Models (LLMs) that generate trivial or hard-coded solutions to pass unit tests rather than producing general, correct code [39]. Krakovna et al. [40] have catalogued many such failure cases across diverse domains. Several studies have analyzed the causes of reward hacking [1, 41, 37], often interpreting it as a manifestation of Goodhart’s Law [42]: when a proxy metric becomes a target for optimization, it ceases to be a good measure. In reinforcement learning, this risk is particularly acute because agents can exploit even small imperfections in the reward specification. Pan et al. [36] further propose a taxonomy of proxy reward misspecification into three types: *misweighting*, *ontological*, and *scope* errors.

To mitigate such risks, several reward-centric methods have been proposed [43, 44]. Inverse Reward Design [43] aims to infer the intended true objective from a given proxy and its training context, helping agents generalize without exploiting flawed signals. Recent work by Rame et al. [44] averages the parameters of multiple reward models to smooth out idiosyncratic errors, reduce the impact of individual proxy biases, and demonstrate reduced reward hacking on held-out tests. Another line of defense focuses on regularizing policy behavior to reduce sensitivity to reward flaws. Common approaches include penalizing divergence from a reference policy using KL-regularization [45]. Recent research by Laidlaw et al. [25] proposes Occupancy-Regularized Policy Optimization (ORPO), which applies a χ^2 penalty on the state-action distribution to constrain deviation from a baseline policy and reduce exploitative behaviors.

Overall, existing approaches either attempt to correct the reward specification or regularize against a fixed proxy. In contrast, our method trains policies against an entire set of plausible proxy rewards, those that remain sufficiently correlated with the true reward, offering robustness to a broader range of misspecifications. Moreover, we show that this robust training objective can be reformulated as an equivalent regularized optimization problem, providing both theoretical and practical benefits.

C.2 Robust Reinforcement Learning

Our work is also related to robust reinforcement learning, where the agent assumes the reward function (and/or transition dynamics) lies within a given uncertainty set, and it seeks to maximize performance against the worst-case realization from that set. This can be formulated as a zero-sum dynamic game between the agent and an adversary who selects the most adverse reward or dynamics; solving the robust MDP thus involves a challenging max-min optimization [46, 47]. To alleviate the computational complexity, early works in this vein rely on a rectangularity assumption that is crucial for traceability. Thus, classical robust RL formulation typically considers rectangular uncertainty sets on rewards or transition probabilities, which lead to conservative solutions but permit efficient algorithms such as robust value iteration [48, 49] or modified policy iteration (MPI) [50].

Recent theoretical work has revealed an intimate connection between adversarial robustness and policy regularization in the context of rectangular uncertainty sets. Several researchers have shown that solving a robust MDP is equivalent to solving a certain regularized RL problem [51, 52, 53]. In particular, the worst-case effect of the adversary can often be captured via an additional penalty term in the agent’s objective. Derman et al. [51] prove that any entropy- or L^2 -regularized MDP can be interpreted as a robust MDP with uncertain rewards – in fact, a regularized MDP is a special case of a reward-robust MDP. Their analysis establishes a duality between a max-min reward-robust objective and a single-agent maximization of expected reward plus a regularization term. Eysenbach and Levine [52] show that the optimal policy from a maximum entropy RL formulation is provably robust to some adversarial reward perturbations. More recently, these insights have been extended and formalized for general MDPs and even multi-agent settings. McMahan et al. [53] study robust Markov games with (s, a) -rectangular uncertainty, and they prove that computing a robust equilibrium is polynomial-time equivalent to computing an equilibrium in a corresponding regularized game. In their framework, the added regularization term is exactly the support function of the uncertainty set, effectively the dual representation of the adversary’s worst-case reward selection. This means that for common uncertainty sets (e.g., those inducing entropy or ℓ_p -norm regularizers), one can replace the inner minimization over rewards with an explicit regularization term in the objective.

The setting in our work departs from the above literature by considering **non-rectangular** reward uncertainty. In particular, we assume a correlation-constrained uncertainty set for the reward function, meaning that the adversary’s permissible deviations in reward are coupled across states. This structure

can mitigate the conservativeness of the worst-case solution (the adversary cannot simultaneously push all state rewards to their extreme worst values) [54], but it also means that the neat robustness-regularization duality from the rectangular-case no longer applies and the robust optimization must be solved (or approximated) directly. In summary, our work tackles a form of reward uncertainty which lies beyond the scope of existing robustness-as-regularization analysis.

D Proofs and Additional Theoretical Results

D.1 Solve the Max-Min Objective

In this section, we show the complete proof for solving the following max-min problem:

$$\max_{\pi} \min_{R \in \mathcal{R}_{\text{corr}}} J(\pi, R) = \max_{\pi} \min_{R \in \mathcal{R}_{\text{corr}}} \mathbb{E}_{(s,a) \sim \mu_{\pi}} [R(s, a)]. \quad (15)$$

where $\mathcal{R}_{\text{corr}}$ is the entire space of rewards that satisfy the correlation constraint with respect to a known proxy reward, as defined below:

$$\begin{aligned} \mathcal{R}_{\text{corr}} &= \left\{ R : (s, a) \rightarrow \mathbb{R} \left| \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{R - M}{V} \cdot R_{\text{proxy}} \right] = r, J(\pi_{\text{ref}}, R) = M, \sigma_R^2 = V^2 \right. \right\} \\ &= \left\{ R : (s, a) \rightarrow \mathbb{R} \left| \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{R - M}{V} \cdot R_{\text{proxy}} \right] = r, \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [R] = M, \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [R^2] = V^2 + M^2 \right. \right\}. \end{aligned} \quad (16)$$

M and V denote the fixed mean and standard deviation of the reward function R under the reference policy π_{ref} . R_{proxy} is the normalized proxy reward

$$R_{\text{proxy}}(s, a) := \frac{\tilde{R}_{\text{proxy}}(s, a) - J(\pi_{\text{ref}}, \tilde{R}_{\text{proxy}})}{\sigma_{\tilde{R}_{\text{proxy}}}}$$

where \tilde{R}_{proxy} is the original (unnormalized) proxy reward. After normalization, we have $J(\pi_{\text{ref}}, R_{\text{proxy}}) = 0$ and $\text{Var}_{\mu_{\pi_{\text{ref}}}}(R_{\text{proxy}}) = 1$.

To solve the challenge that the objective $\mathbb{E}_{\mu_{\pi}} [R(s, a)]$ depends on the state-action occupancy μ_{π} , whereas the constraints defining $\mathcal{R}_{\text{corr}}$ are expressed in terms of $\mu_{\pi_{\text{ref}}}$. We apply a *change-of-measure* technique [28, 29] to rewrite the expectation under $\mu_{\pi_{\text{ref}}}$. Specifically, let $L(s, a)$ denote the Radon-Nikodym derivative:

$$L(s, a) = \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$$

By definition, $L(s, a) \geq 0$ and $\mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L(s, a)] = 1$. Applying the change-of-measure formula, we can express the return as:

$$\begin{aligned} \mathbb{E}_{\mu_{\pi}} [R(s, a)] &= \int_{\mathcal{S} \times \mathcal{A}} \mu_{\pi}(s, a) R(s, a) d(s, a) \\ &= \int_{\mathcal{S} \times \mathcal{A}} \mu_{\pi_{\text{ref}}}(s, a) \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)} R(s, a) d(s, a) \\ &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L(s, a) R(s, a)] \end{aligned}$$

Thus, both the objective and the constraints can be rewritten as expectations with respect to the reference distribution $\mu_{\pi_{\text{ref}}}$. For notational simplicity, we will suppress the variables (s, a) when necessary. Under this reparameterization, the max-min objective in Equation 15 can be reformulated as:

$$\max_{\pi} \min_{R \in \mathcal{R}_{\text{corr}}} \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L \cdot R]. \quad (17)$$

Solve Inner Minimization Problem. The Lagrangian functional associated with the inner minimization problem of 17 is defined as:

$$l_0(\lambda_1, \lambda_2, \lambda_3, R) = \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[L \cdot R - \lambda_1 \frac{R - M}{V} \cdot R_{\text{proxy}} - \lambda_2 R - \lambda_3 R^2 \right] + \lambda_1 r + \lambda_2 M + \lambda_3 (M^2 + V^2)$$

1012 where $\lambda_1, \lambda_2, \lambda_3$ are the Lagrange multipliers corresponding to the correlation constraint, mean
 1013 constraint, and variance constraint, respectively. Then the inner minimization problem in Equation 17
 1014 is equivalent to the following problem:

$$\max_{\lambda_1, \lambda_2, \lambda_3} \min_{R \in \mathcal{R}_{\text{corr}}} l_0(\lambda_1, \lambda_2, \lambda_3, R) \quad (18)$$

1015 We now solve the inner minimization problem in Equation 18 by finding the optimal R for fixed dual
 1016 variables $(\lambda_1, \lambda_2, \lambda_3)$. Taking the functional derivative of the Lagrangian l_0 with respect to $R(s, a)$
 1017 gives:

$$\frac{\partial l_0}{\partial R} = \mu_{\pi_{\text{ref}}}(s, a) \left[\left(L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2 \right) - 2\lambda_3 R \right]$$

1018 When $\mu_{\pi_{\text{ref}}}(s, a) > 0$, setting the derivative of the Lagrangian to zero yields the optimal adversarial
 1019 reward function:

$$R^*(s, a) = \frac{L(s, a) - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2}{2\lambda_3} \quad (19)$$

1020 However, for state-action pairs where $\mu_{\pi_{\text{ref}}}(s, a) = 0$, i.e., those not visited under the reference policy,
 1021 the correlation and moment constraints become vacuous. In these regions, the adversarial reward
 1022 $R^*(s, a)$ can be driven arbitrarily poor, reflecting that no constraint prevents the adversary from
 1023 assigning highly penalizing values to rarely visited or unobserved state-action pairs. Nevertheless,
 1024 consider the case where $\mu_{\pi_{\text{ref}}}(s, a) > 0$, after substituting the optimal R^* from Equation 19 into the
 1025 Lagrangian l_0 in Equation 18 and simplifying, we obtain the following dual objective:

$$\max_{\lambda_1, \lambda_2, \lambda_3} l_0(\lambda_1, \lambda_2, \lambda_3, R^*) = \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{\left(L(s, a) - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2 \right)^2}{4\lambda_3} \right] + \lambda_1 r + \lambda_2 M + \lambda_3 (M^2 + V^2) \quad (20)$$

1026 We now compute the gradients of the dual objective with respect to the dual variables:

$$\begin{aligned} \frac{\partial l_0}{\partial \lambda_1} &= -\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{\left(L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2 \right) \frac{R_{\text{proxy}}(s, a)}{V}}{2\lambda_3} \right] + r \\ \frac{\partial l_0}{\partial \lambda_2} &= -\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2}{2\lambda_3} \right] + M \\ \frac{\partial l_0}{\partial \lambda_3} &= -\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{\left(L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2 \right)^2}{4\lambda_3^2} \right] + M^2 + V^2 \end{aligned} \quad (21)$$

1027 Setting these gradients to zero yields the system of equations:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{\left(L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2 \right) \frac{R_{\text{proxy}}(s, a)}{V}}{2\lambda_3} \right] = r, \quad (22)$$

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2}{2\lambda_3} \right] = M, \quad (23)$$

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{\left(L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2 \right)^2}{4\lambda_3^2} \right] = M^2 + V^2. \quad (24)$$

1028 Expanding and simplifying each condition:

1029 **Solving for λ_2 :** Starting with Equation 23,

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L] - \lambda_1 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\frac{R_{\text{proxy}}(s, a)}{V}\right] - \lambda_2 = 2\lambda_3 M$$

1030 Recall from our normalization that $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}] = 0$. Thus,

$$\lambda_2 = \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L] - 2\lambda_3 M$$

1031 Since $L(s, a) = \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$, and using properties of Radon-Nikodym derivatives, we have:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L] = 1$$

1032 Thus, we find:

$$\boxed{\lambda_2 = 1 - 2\lambda_3 M}$$

1033 **Solving for λ_1 :** Substituting $\lambda_2 = 1 - 2\lambda_3 M$ into Equation 22,

$$\begin{aligned} 2r\lambda_3 &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\left(L - \lambda_1 \frac{R_{\text{proxy}}}{V} - 1 + 2\lambda_3 M\right) \frac{R_{\text{proxy}}}{V}\right] \\ &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[L \cdot \frac{R_{\text{proxy}}}{V}\right] - \lambda_1 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\frac{R_{\text{proxy}}^2}{V^2}\right] - (1 - 2\lambda_3 M) \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\frac{R_{\text{proxy}}}{V}\right] \end{aligned}$$

1034 Again, using normalization, $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}] = 0$ and $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}^2] = 1$, so we get:

$$2r\lambda_3 = \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[L \cdot \frac{R_{\text{proxy}}}{V}\right] - \frac{\lambda_1}{V^2}$$

1035 which rearranges to:

$$\boxed{\lambda_1 = V \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R_{\text{proxy}}] - 2r\lambda_3 V^2}$$

1036 **Solving for λ_3 :** Substituting $\lambda_2 = 1 - 2\lambda_3 M$ into Equation 24,

$$\begin{aligned} 4\lambda_3^2(M^2 + V^2) &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\left(L - \lambda_1 \frac{R_{\text{proxy}}}{V} - 1 + 2\lambda_3 M\right)^2\right] \\ &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] + \lambda_1^2 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\frac{R_{\text{proxy}}^2}{V^2}\right] + (1 - 2\lambda_3 M)^2 - 2\lambda_1 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[L \cdot \frac{R_{\text{proxy}}}{V}\right] \\ &\quad - 2(1 - 2\lambda_3 M) \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L] + 2\lambda_1(1 - 2\lambda_3 M) \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\frac{R_{\text{proxy}}}{V}\right] \end{aligned}$$

1037 Again, using normalization ($\mathbb{E}[R_{\text{proxy}}] = 0$, $\mathbb{E}[R_{\text{proxy}}^2] = 1$, $\mathbb{E}[L] = 1$), this simplifies to:

$$\begin{aligned} 4\lambda_3^2(M^2 + V^2) &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] + \frac{\lambda_1^2}{V^2} - 2\lambda_1 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[L \cdot \frac{R_{\text{proxy}}}{V}\right] + 4\lambda_3^2 M^2 - 1 \\ 4\lambda_3^2 V^2 &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] + \frac{\lambda_1^2}{V^2} - 2\lambda_1 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[L \cdot \frac{R_{\text{proxy}}}{V}\right] - 1 \end{aligned}$$

1038 Now substitute $\lambda_1 = V \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R_{\text{proxy}}] - 2r\lambda_3 V^2$ into this expression. After rearrangement and
1039 simplification, we obtain:

$$4\lambda_3^2(1 - r^2)V^2 = \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] - \mathbb{E}_{\mu_{\pi_{\text{ref}}}}^2[L \cdot R_{\text{proxy}}] - 1$$

1040 Thus,

$$\lambda_3 = \pm \frac{1}{2} \frac{\sqrt{\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] - \mathbb{E}_{\mu_{\pi_{\text{ref}}}}^2[L \cdot R_{\text{proxy}}] - 1}}{V \sqrt{1 - r^2}}$$

1041 We argue that $\lambda_3 < 0$ yields the optimal dual variable. To determine which root maximizes the above
1042 dual objective in Equation 20, we compute the second derivative from Equation 21:

$$\frac{\partial^2 l_0}{\partial \lambda_3^2} = \mathbb{E}_{\mu_{\pi_{\text{ref}}}}\left[\frac{\left(L - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2\right)^2}{2\lambda_3^3}\right]$$

1043 Since the numerator is always non-negative and when $\lambda_3 < 0$, we have $\frac{\partial^2 l_0}{\partial \lambda_3^2} < 0$, which implies
 1044 the dual objective is concave in λ_3 around this root. Thus, selecting the negative root yields a local
 1045 maximum of the dual objective.

1046 Recognizing that $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] - 1$ corresponds to the χ^2 divergence between the occupancy measures:

$$\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) = \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] - 1$$

1047 and noting that:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R_{\text{proxy}}] = \mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}]$$

1048 we can express the solution for λ_3 as:

$$\lambda_3 = -\frac{\sqrt{\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) - \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]}}{2V\sqrt{1-r^2}}. \quad (25)$$

1049 **Solve Outer Maximization Problem.** Now that we have solved for the optimal primal variable
 1050 R and dual variables λ_1 , λ_2 , and λ_3 , we plug them back into the original max-min objective in
 1051 Equation 17:

$$\max_{\pi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R] = \max_{\pi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[L(s, a) \cdot \frac{L(s, a) - \lambda_1 \frac{R_{\text{proxy}}(s, a)}{V} - \lambda_2}{2\lambda_3} \right] \quad (26)$$

1052 Using the earlier substitutions:

$$\begin{aligned} \lambda_1 &= V \cdot \mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] - 2r\lambda_3V^2, \\ \lambda_2 &= 1 - 2\lambda_3M, \\ \lambda_3 &= -\frac{1}{2} \cdot \frac{\sqrt{\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) - \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]}}{V\sqrt{1-r^2}}, \end{aligned}$$

1053 We simplify the expression:

$$\max_{\pi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[LR] = \max_{\pi} \frac{1}{2\lambda_3} \left(\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] - \lambda_1 \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[L \cdot \frac{R_{\text{proxy}}}{V} \right] - \lambda_2 \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L] \right)$$

1054 Recall the identities:

$$\begin{aligned} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L^2] &= \chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) + 1, \\ \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R_{\text{proxy}}] &= \mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}], \\ \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L] &= 1, \end{aligned}$$

1055 We substitute these and get:

$$\max_{\pi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[LR] = \frac{1}{2\lambda_3} \left(\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) + 1 - \lambda_1 \cdot \frac{\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}]}{V} - \lambda_2 \right)$$

1056 Now substitute the expressions for λ_1 and λ_2 :

$$\begin{aligned} \max_{\pi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[LR] &= \max_{\pi} \frac{1}{2\lambda_3} \left(\chi^2 + 1 - (\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] - 2r\lambda_3V) \cdot \frac{\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}]}{V} - (1 - 2\lambda_3M) \right) \\ &= \max_{\pi} \frac{1}{2\lambda_3} \left(\chi^2 - \frac{\mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]}{V} + 2r\lambda_3\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] + 2\lambda_3M \right) \end{aligned}$$

1057 Now cancel out $2\lambda_3$ in numerator and denominator:

$$\max_{\pi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[LR] = \max_{\pi} \mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] \cdot r - \frac{1}{2\lambda_3} \cdot \left(\frac{\mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]}{V} - \chi^2 \right) + M$$

1058 Now plug in the expression for λ_3 :

$$\lambda_3 = -\frac{1}{2} \cdot \frac{\sqrt{\chi^2 - \mathbb{E}_{\mu_\pi}^2[R_{\text{proxy}}]}}{V\sqrt{1-r^2}}$$

1059 This gives the final outer problem for the original max-min objective in Equation 15:

$$\max_{\pi} \quad r \cdot V \cdot \mathbb{E}_{\mu_\pi}[R_{\text{proxy}}] - V \cdot \sqrt{1-r^2} \cdot \sqrt{\chi^2(\mu_\pi \parallel \mu_{\pi_{\text{ref}}}) - \mathbb{E}_{\mu_\pi}^2[R_{\text{proxy}}]} + M \quad (27)$$

1060 D.2 Proof of Optimality

1061 Recall the inner minimization problem of our max-min objective in Equation 17:

$$\min_{R \in \mathcal{R}_{\text{corr}}} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R]$$

1062 where $L = \mu_\pi(s, a) / \mu_{\pi_{\text{ref}}}(s, a)$ is treated as fixed, and the feasible set is:

$$\mathcal{R}_{\text{corr}} = \left\{ R : (s, a) \rightarrow \mathbb{R} \left| \begin{array}{l} \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[(R - M) \cdot R_{\text{proxy}}] = r \cdot V, \\ \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R] = M, \quad \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R^2] = V^2 + M^2 \end{array} \right. \right\}$$

1063 The feasible region is not convex due to the *quadratic equality* constraint $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R^2] = V^2 + M^2$.

1064 This defines the boundary of an L^2 ball (a hypersphere) in function space, which is not convex.
1065 Therefore, traditional convex programming tools and strong duality do not directly apply.

1066 However, we still claim that the resulting R^* derived in Appendix D.1 is globally optimal. This is
1067 supported by the following facts:

1068 **Stationarity.** When considering R^* for any fixed dual variables $\lambda_1, \lambda_2, \lambda_3$, we are looking at the
1069 inner minimization problem in Equation 18 as follows:

$$\min_{R \in \mathcal{R}_{\text{corr}}} l_0(\lambda_1, \lambda_2, \lambda_3, R)$$

1070 The term with $R(s, a)$ in l_0 is:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R - \lambda_1 \frac{R - M}{V} \cdot R_{\text{proxy}} - \lambda_2 R - \lambda_3 R^2]$$

1071 For this quadratic in R to have a minimum (since it is a minimization problem for R), the coefficient
1072 of R^2 must be positive. In our case, the coefficient is $-\lambda_3$. Therefore, for the minimization problem
1073 to be well-posed and have a finite minimum, we must have $\lambda_3 < 0$. This condition ensures that the
1074 quadratic term in R is a concave upward parabola, which means that a minimum exists. Moreover, in
1075 Appendix D.1, we explicitly state that $R^*(s, a)$ is derived by setting the derivative of the Lagrangian
1076 function l_0 in Equation 18 to zero with respect to $R(s, a)$. Thus, $R^*(s, a)$ is indeed the optimal value
1077 for the minimization problem for fixed $\lambda_1, \lambda_2, \lambda_3$ where $\lambda_3 < 0$. The Stationarity in this context
1078 implies that R^* lies within the domain where the Lagrangian is well-defined and differentiable, which
1079 it does.

1080 **Feasibility.** We also argue that the closed-form primal solution $R^*(\lambda^*)$, where λ^* denotes the
1081 optimal dual solution, is feasible in the original sense, that is, it satisfies the three equality constraints
1082 in the feasible set $\mathcal{R}_{\text{corr}}$. Specifically, as detailed in Appendix D.1, we substitute R^* back into the
1083 dual objective l_0 and compute the gradient with respect to each dual variable. We then solve:

$$\frac{\partial l_0(\lambda_1, \lambda_2, \lambda_3, R^*(\lambda_1, \lambda_2, \lambda_3))}{\partial \lambda_i} = 0, \quad \text{for } i = 1, 2, 3,$$

1084 to find the optimal values $\lambda_1^*, \lambda_2^*, \lambda_3^*$.

1085 By the chain rule, we have:

$$\frac{\partial l_0(\lambda_1, \lambda_2, \lambda_3, R^*(\lambda_1, \lambda_2, \lambda_3))}{\partial \lambda_i} = \left\langle \frac{\partial l_0}{\partial R}, \frac{\partial R^*}{\partial \lambda_i} \right\rangle + \frac{\partial l_0}{\partial \lambda_i},$$

1086 where the first term vanishes because R^* is chosen to minimize l_0 for fixed λ (i.e., $\partial l_0 / \partial R = 0$ at
1087 R^*). Therefore, the derivative simplifies to:

$$\frac{\partial l_0(\lambda_1, \lambda_2, \lambda_3, R^*)}{\partial \lambda_i} = \frac{\partial l_0}{\partial \lambda_i}.$$

1088 Setting these derivatives to zero yields:

$$\begin{aligned} \frac{\partial l_0}{\partial \lambda_1} &= -\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[(R^* - M)R_{\text{proxy}}] + rV = 0, \\ \frac{\partial l_0}{\partial \lambda_2} &= -\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R^*] + M = 0, \\ \frac{\partial l_0}{\partial \lambda_3} &= -\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[(R^*)^2] + V^2 + M^2 = 0, \end{aligned}$$

1089 which exactly recover the original feasibility constraints. Hence, the solution $R^*(\lambda^*)$ is feasible by
1090 construction.

1091 Therefore, R^* satisfies both stationarity and feasibility, which is sufficient for global optimality.

1092 **D.3 Proof that $\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) \geq \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]$**

1093 To ensure that the inner term of the square root in Equation 25 remains non-negative, we need to
1094 show that

$$\chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}}) \geq \mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]$$

1095 **Proof.** Recall that

$$\mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] = \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[L \cdot R_{\text{proxy}}],$$

1096 where $L(s, a) = \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$ is the Radon-Nikodym derivative. Since R_{proxy} is normalized to have zero
1097 mean under $\mu_{\pi_{\text{ref}}}$, we have:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}] = 0.$$

1098 Thus,

$$\begin{aligned} \mathbb{E}_{\mu_{\pi}}[R_{\text{proxy}}] &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}(s, a)(L(s, a) - 1)] \\ &= \sum_{(s, a)} R_{\text{proxy}}(s, a) \mu_{\pi_{\text{ref}}}(s, a) (L(s, a) - 1) \end{aligned}$$

1099 Applying the Cauchy-Schwarz inequality:

$$\left(\sum_{(s, a)} R_{\text{proxy}}(s, a) \mu_{\pi_{\text{ref}}}(s, a) (L(s, a) - 1) \right)^2 \leq \left(\sum_{(s, a)} \mu_{\pi_{\text{ref}}}(s, a) R_{\text{proxy}}^2(s, a) \right) \left(\sum_{(s, a)} \mu_{\pi_{\text{ref}}}(s, a) (L(s, a) - 1)^2 \right)$$

1100 By the assumptions: $\mathbb{E}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}^2] = 1$, $\sum_{(s, a)} \mu_{\pi_{\text{ref}}}(s, a) (L(s, a) - 1)^2 = \chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}})$.

1101 We obtain:

$$\mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}] \leq \chi^2(\mu_{\pi} \parallel \mu_{\pi_{\text{ref}}})$$

1102 as desired.

1103 D.4 Derive Lagrangian Functional for Linear Max-Min Objective

1104 Recall that our max-min optimization under the structured reward assumption is as follows:

$$\max_{\pi} \min_{\theta \in \mathcal{R}_{\text{corr}}^{\text{lin}}, \theta \geq 0} \mathbb{E}_{(s,a) \sim \mu_{\pi}} [\theta^{\top} \phi(s, a)]. \quad (28)$$

1105 where $\mathcal{R}_{\text{corr}}^{\text{lin}}$ is the uncertainty set defined as follow:

$$\mathcal{R}_{\text{corr}}^{\text{lin}} = \left\{ \theta \in \mathbb{R}^M \mid \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [\theta^{\top} \phi \cdot R_{\text{proxy}}] = r, \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [\theta^{\top} \phi] = 0, \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [(\theta^{\top} \phi)^2] = 1 \right\}. \quad (29)$$

1106 We assume without loss of generality that the worst-case reward $R(s, a) = \theta^{\top} \phi(s, a)$ is normalized
 1107 to have zero mean and unit variance under the reference policy π_{ref} . This corresponds to setting
 1108 $M = 0$ and $V = 1$, which, as shown in our earlier derivation, does not affect the resulting optimal
 1109 policy. As before, R_{proxy} denotes the normalized proxy reward under π_{ref} , satisfying $\mathbb{E}_{\mu_{\pi_{\text{ref}}}} [R_{\text{proxy}}] = 0$
 1110 and $\text{Var}_{\mu_{\pi_{\text{ref}}}} [R_{\text{proxy}}] = 1$.

1111 Similar to previous steps, we introduce the Radon-Nikodym derivative

$$L(s, a) = \frac{\mu_{\pi}(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)}$$

1112 We use a change-of-measure, and define the Lagrangian functional for the inner minimization in
 1113 Equation 28 as:

$$\begin{aligned} l_1(\lambda_1, \lambda_2, \lambda_3, \theta) &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [L \cdot \theta^{\top} \phi] - \lambda_1 \left(\mathbb{E}_{\mu_{\pi_{\text{ref}}}} [R_{\text{proxy}} \cdot \theta^{\top} \phi] - r \right) \\ &\quad - \lambda_2 \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [\theta^{\top} \phi] - \lambda_3 \left(\mathbb{E}_{\mu_{\pi_{\text{ref}}}} [(\theta^{\top} \phi)^2] - 1 \right) \\ &= \mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[(L - \lambda_1 R_{\text{proxy}} - \lambda_2) \theta^{\top} \phi - \lambda_3 \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [(\theta^{\top} \phi)^2] + \lambda_1 r + \lambda_3 \right] \\ &= \sum_{(s,a)} \mu_{\pi_{\text{ref}}}(s, a) \left[(L - \lambda_1 R_{\text{proxy}} - \lambda_2) \theta^{\top} \phi - (\theta^{\top} \phi)^2 \right] + \lambda_1 r + \lambda_3 \end{aligned}$$

1114 Define the following terms for simplicity:

$$\begin{aligned} v(s, a) &= \mu_{\pi}(s, a) \\ D(s, a) &= \mu_{\pi_{\text{ref}}}(s, a) \cdot R_{\text{proxy}}(s, a) \\ C(s, a) &= \mu_{\pi_{\text{ref}}}(s, a) \\ u_{\lambda_1, \lambda_2}(s, a) &= v(s, a) - \lambda_1 D(s, a) - \lambda_2 C(s, a) \end{aligned}$$

1115 Then the Lagrangian function simplifies to:

$$\begin{aligned} l_1(\lambda_1, \lambda_2, \lambda_3, \theta) &= \sum_{(s,a)} \left[u_{\lambda_1, \lambda_2}(s, a) \theta^{\top} \phi(s, a) - \lambda_3 C(s, a) (\theta^{\top} \phi(s, a))^2 \right] + \lambda_1 r + \lambda_3 \\ &= \theta^{\top} \left(\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s, a) \phi(s, a) \right) - \lambda_3 \theta^{\top} \left(\sum_{(s,a)} C(s, a) \phi(s, a) \phi(s, a)^{\top} \right) \theta + \lambda_1 r + \lambda_3 \end{aligned}$$

1116 where we expand the quadratic term:

$$\begin{aligned} \sum_{(s,a)} C(s, a) (\theta^{\top} \phi(s, a))^2 &= \sum_{(s,a)} C(s, a) (\phi(s, a)^{\top} \theta)^2 \\ &= \sum_{(s,a)} C(s, a) \theta^{\top} \phi(s, a) \phi(s, a)^{\top} \theta \\ &= \theta^{\top} \left(\sum_{(s,a)} C(s, a) \phi(s, a) \phi(s, a)^{\top} \right) \theta \end{aligned}$$

1117 Let

$$Q = \sum_{(s,a)} C(s,a) \phi(s,a) \phi(s,a)^\top \quad (30)$$

1118 then we can write the Lagrangian function as:

$$l_1(\lambda_1, \lambda_2, \lambda_3, \theta) = \theta^\top \left(\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s,a) \phi(s,a) \right) - \lambda_3 \theta^\top Q \theta + \lambda_1 r + \lambda_3$$

1119 And the inner minimization problem in Equation 28 becomes:

$$\max_{\lambda_1, \lambda_2, \lambda_3} \min_{\theta \geq 0} l_1(\lambda_1, \lambda_2, \lambda_3, R) \quad (31)$$

1120 D.5 Proof for Whitening Transformation

1121 To simplify the problem associated with the Lagrangian function above, we transform the feature
1122 vector ϕ into a whitened version $\tilde{\phi}$ such that the matrix Q as defined in Equation 30 becomes
1123 the identity matrix I . Specifically, we perform a whitening transformation using the Cholesky
1124 decomposition [32]. Let

$$W = Q^{-\frac{1}{2}}, \quad \tilde{\phi}(s,a) = W \phi(s,a)$$

1125 where $Q^{-\frac{1}{2}}$ denotes a matrix square root of Q^{-1} (which exists since Q is positive semi-definite and
1126 non-singular assuming $\exists(s,a)$ such that $C(s,a) > 0$).

1127 Then:

$$\begin{aligned} \sum_{(s,a)} C(s,a) \tilde{\phi}(s,a) \tilde{\phi}(s,a)^\top &= \sum_{(s,a)} C(s,a) (W \phi(s,a)) (W \phi(s,a))^\top \\ &= \sum_{(s,a)} C(s,a) W \phi(s,a) \phi(s,a)^\top W^\top \\ &= W \left(\sum_{(s,a)} C(s,a) \phi(s,a) \phi(s,a)^\top \right) W^\top \\ &= W Q W^\top \\ &= Q^{-\frac{1}{2}} Q Q^{-\frac{1}{2}} \\ &= I \end{aligned}$$

1128 as desired.

1129 Since $C(s,a) = \mu_{\pi_{\text{ref}}}(s,a) \geq 0$ by definition of the discounted occupancy measure, and assuming
1130 there exists at least one (s,a) with $C(s,a) > 0$, the matrix Q is positive definite, and the whitening
1131 transformation is well-defined.

1132 D.6 Derive Optimal Primal Variable for Linear Max-Min Objective

1133 After whitening transformation as discussed in Appendix D.5, the problem in Equation 31 becomes:

$$\max_{\lambda_1, \lambda_2, \lambda_3} \min_{\tilde{\theta} \geq 0} l_1(\lambda_1, \lambda_2, \lambda_3, \tilde{\theta}) = \tilde{\theta}^\top \left(\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s,a) \tilde{\phi}(s,a) \right) - \lambda_3 \tilde{\theta}^\top \tilde{\theta} + \lambda_1 r + \lambda_3. \quad (32)$$

1134 where we now optimize over the parameter $\tilde{\theta}$ using the transformed features $\tilde{\phi}$. For notational
1135 simplicity, we will drop the tilde and henceforth use ϕ to represent the whitened feature $\tilde{\phi}$, and θ to
1136 represent the whitened weights $\tilde{\theta}$.

1137 **Separable Structure.** In the whitened feature space, the objective becomes separable across
 1138 coordinates of θ . Thus, the inner minimization problem in Equation 32 decouples into M independent
 1139 one-dimensional convex minimization problems, one for each feature coordinate $i \in \{1, 2, \dots, M\}$:

$$\min_{\theta_i \geq 0} \left(\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s, a) \phi_i(s, a) \right) \theta_i - \lambda_3 \theta_i^2$$

1140 Let us solve the i -th subproblem. Assuming $\lambda_3 < 0$, the objective is a convex quadratic function in
 1141 θ_i (an upward-opening parabola). The unconstrained minimum occurs at:

$$\theta_i^* = - \frac{\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s, a) \phi_i(s, a)}{2\lambda_3}$$

1142 Considering the constraint $\theta_i \geq 0$, we have two cases:

- 1143 • If the unconstrained minimum $\theta_i^* \geq 0$, then it is also the solution to the constrained problem.
- 1144 • If $\theta_i^* < 0$, then the constrained minimum occurs at the boundary $\theta_i = 0$.

1145 Thus, the final optimal θ_i^* is:

$$\theta_i^* = \max \left(0, - \frac{\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s, a) \phi_i(s, a)}{2\lambda_3} \right)$$

1146 Collecting across all i , we express the final optimal solution θ^* as:

$$\theta^* = \max \left(0, - \frac{\sum_{(s,a)} u_{\lambda_1, \lambda_2}(s, a) \phi(s, a)}{2\lambda_3} \right) \quad (33)$$

1147 where the $\max(\cdot, 0)$ is applied elementwise.

1148 D.7 Solve the Dual Objective for Linear Max-Min Objective

1149 Let the outer objective in Equation 32 be:

$$g(\lambda_1, \lambda_2, \lambda_3) = l_1(\lambda_1, \lambda_2, \lambda_3, \theta^*)$$

1150 Then we want to solve the following dual objective:

$$\max_{\lambda_1, \lambda_2, \lambda_3} g(\lambda_1, \lambda_2, \lambda_3) \quad (34)$$

1151 Let

$$q_j(\lambda_1, \lambda_2) = \sum_{(s,a)} (v(s, a) - \lambda_1 D(s, a) - \lambda_2 C(s, a)) \phi_j(s, a)$$

1152 denote the linear coefficient for each feature $j \in \{1, \dots, M\}$.

1153 The optimal θ_j^* is:

$$\theta_j^*(\lambda) = \max \left(0, \frac{q_j(\lambda_1, \lambda_2)}{2\lambda_3} \right)$$

1154 Now, we compute the gradients:

Gradient with respect to λ_1 :

$$\begin{aligned} \frac{\partial g}{\partial \lambda_1}(\lambda) &= \frac{\partial l_1}{\partial \lambda_1}(\lambda, \theta^*(\lambda)) \\ &= \sum_{(s,a)} \left(-D(s, a) (\theta^{*T} \phi(s, a)) \right) + r \\ &= r - \sum_{j=1}^M D_{\phi, j} \cdot \theta_j^*(\lambda) \end{aligned}$$

1155 where

$$D_{\phi, j} = \sum_{(s,a)} D(s, a) \phi_j(s, a)$$

Gradient with respect to λ_2 :

$$\begin{aligned}\frac{\partial g}{\partial \lambda_2}(\lambda) &= \frac{\partial l_1}{\partial \lambda_2}(\lambda, \boldsymbol{\theta}^*(\lambda)) \\ &= \sum_{(s,a)} \left(-C(s,a) (\boldsymbol{\theta}^{*T} \boldsymbol{\phi}(s,a)) \right) \\ &= - \sum_{j=1}^M C_{\phi,j} \cdot \theta_j^*(\lambda)\end{aligned}$$

1156 where

$$C_{\phi,j} = \sum_{(s,a)} C(s,a) \phi_j(s,a)$$

Gradient with respect to λ_3 :

$$\begin{aligned}\frac{\partial g}{\partial \lambda_3}(\lambda) &= \frac{\partial l_1}{\partial \lambda_3}(\lambda, \boldsymbol{\theta}^*(\lambda)) \\ &= \sum_{(s,a)} \left(-C(s,a) (\boldsymbol{\theta}^{*T} \boldsymbol{\phi}(s,a))^2 \right) + 1 \\ &= 1 - \sum_{j=1}^M (\theta_j^*(\lambda))^2\end{aligned}$$

1157 where we use the whitening assumption $\sum_{(s,a)} C(s,a) \boldsymbol{\phi}(s,a) \boldsymbol{\phi}(s,a)^\top = I$.

1158 Thus, the full gradients are:

$$\boxed{\begin{aligned}\frac{\partial g}{\partial \lambda_1}(\lambda) &= r - \sum_{j=1}^M D_{\phi,j} \cdot \max \left(0, \frac{q_j(\lambda_1, \lambda_2)}{2\lambda_3} \right) \\ \frac{\partial g}{\partial \lambda_2}(\lambda) &= - \sum_{j=1}^M C_{\phi,j} \cdot \max \left(0, \frac{q_j(\lambda_1, \lambda_2)}{2\lambda_3} \right) \\ \frac{\partial g}{\partial \lambda_3}(\lambda) &= 1 - \sum_{j=1}^M \left(\max \left(0, \frac{q_j(\lambda_1, \lambda_2)}{2\lambda_3} \right) \right)^2\end{aligned}}$$

1159 We can solve for the optimal dual variables $(\lambda_1, \lambda_2, \lambda_3)$ using standard first-order optimization
1160 methods. Since $g(\lambda)$ is concave (under the condition $\lambda_3 < 0$), optimization is well-behaved and
1161 converges reliably. After obtaining the optimal primal variables $\boldsymbol{\theta}^*$ and dual variables $(\lambda_1^*, \lambda_2^*, \lambda_3^*)$,
1162 we can substitute them back into Equation 28 and solve the outer maximization over the policy π
1163 using standard reinforcement learning algorithms, such as PPO [33].

1164 D.8 Policy Gradient Derivation

1165 We now derive the gradient of the robust objective (27) with respect to the policy parameters θ . Recall
1166 that the robust objective is:

$$\begin{aligned}\mathcal{J}(\mu_{\pi_\theta}) &= r \cdot V \cdot \mathbb{E}_{\mu_{\pi_\theta}}[R_{\text{proxy}}] - V \cdot \sqrt{1-r^2} \cdot \sqrt{\chi^2(\mu_{\pi_\theta} \parallel \mu_{\pi_{\text{ref}}}) - \left(\mathbb{E}_{\mu_{\pi_\theta}}[R_{\text{proxy}}] \right)^2} + M \\ &= \mathbb{E}_{\mu_{\pi_\theta}}[R_{\text{proxy}}] - \frac{\sqrt{1-r^2}}{r} \sqrt{\chi^2(\mu_{\pi_\theta} \parallel \mu_{\pi_{\text{ref}}}) - \left(\mathbb{E}_{\mu_{\pi_\theta}}[R_{\text{proxy}}] \right)^2}\end{aligned}$$

1167 where we set $M = 0$ and $V = 1$ without loss of generality. We also divide the entire objective by r ,
1168 which is assumed to be positive ($r > 0$), so this rescaling preserves the optimization direction and

1169 does not affect the final policy solution. The χ^2 divergence is defined as:

$$\chi^2(\mu_{\pi_\theta} \parallel \mu_{\pi_{\text{ref}}}) = \sum_{(s,a)} \frac{\mu_{\pi_\theta}(s,a)^2}{\mu_{\pi_{\text{ref}}}(s,a)} - 1$$

1170 Applying the chain rule, we compute:

$$\nabla_\theta \mathcal{J} = \nabla_\theta \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] - \frac{\sqrt{1-r^2}}{r} \nabla_\theta \left(\sqrt{h(\mu_{\pi_\theta})} \right) \quad (35)$$

1171 where we define:

$$h(\mu_{\pi_\theta}) = \chi^2(\mu_{\pi_\theta} \parallel \mu_{\pi_{\text{ref}}}) - \left(\mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] \right)^2$$

1172 Using the chain rule again:

$$\nabla_\theta \sqrt{h(\mu_{\pi_\theta})} = \frac{1}{2\sqrt{h(\mu_{\pi_\theta})}} \nabla_\theta h(\mu_{\pi_\theta})$$

1173 Now compute $\nabla_\theta h(\mu_{\pi_\theta})$:

$$\begin{aligned} \nabla_\theta h(\mu_{\pi_\theta}) &= \nabla_\theta \chi^2(\mu_{\pi_\theta} \parallel \mu_{\pi_{\text{ref}}}) - \nabla_\theta \left(\mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}]^2 \right) \\ &= \nabla_\theta \left(\sum_{(s,a)} \frac{\mu_{\pi_\theta}(s,a)^2}{\mu_{\pi_{\text{ref}}}(s,a)} - 1 \right) - 2 \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] \nabla_\theta \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] \end{aligned}$$

1174 The individual terms are:

$$\begin{aligned} \nabla_\theta \left(\sum_{(s,a)} \frac{\mu_{\pi_\theta}(s,a)^2}{\mu_{\pi_{\text{ref}}}(s,a)} - 1 \right) &= 2 \sum_{(s,a)} \frac{\mu_{\pi_\theta}(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) \\ \nabla_\theta \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] &= \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) R_{\text{proxy}}(s,a) \end{aligned}$$

1175 Thus:

$$\nabla_\theta h(\mu_{\pi_\theta}) = \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) \left(2 \frac{\mu_{\pi_\theta}(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)} - 2 \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] R_{\text{proxy}}(s,a) \right)$$

1176 Then we compute $\nabla_\theta \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}]$:

$$\begin{aligned} \nabla_\theta \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] &= \nabla_\theta \sum_{(s,a)} \mu_{\pi_\theta}(s,a) R_{\text{proxy}}(s,a) \\ &= \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) R_{\text{proxy}}(s,a) \end{aligned}$$

1177 Put them together, we get the final gradient in Equation 35 as:

$$\begin{aligned} \nabla_\theta \mathcal{J} &= \nabla_\theta \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] - \frac{\sqrt{1-r^2}}{r} \nabla_\theta \left(\sqrt{h(\mu_{\pi_\theta})} \right) \\ &= \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) R_{\text{proxy}}(s,a) - \frac{\sqrt{1-r^2}}{r} \frac{1}{2\sqrt{h(\mu_{\pi_\theta})}} \nabla_\theta h(\mu_{\pi_\theta}) \\ &= \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) R_{\text{proxy}}(s,a) \\ &\quad - \frac{\sqrt{1-r^2}}{r} \frac{1}{2\sqrt{h(\mu_{\pi_\theta})}} \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) \left(2 \frac{\mu_{\pi_\theta}(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)} - 2 \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] R_{\text{proxy}}(s,a) \right) \\ &= \sum_{(s,a)} \nabla_\theta \mu_{\pi_\theta}(s,a) \left[R_{\text{proxy}} - \frac{\sqrt{1-r^2}}{r} \frac{1}{\sqrt{h(\mu_{\pi_\theta})}} \left(\frac{\mu_{\pi_\theta}(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)} - \mathbb{E}_{\mu_{\pi_\theta}} [R_{\text{proxy}}] R_{\text{proxy}}(s,a) \right) \right] \end{aligned} \quad (36)$$

1178 The full policy gradient for the ORPO algorithm, as presented in Appendix B of [25], is given by:

$$\sum_{(s,a)} \nabla_{\theta} \mu_{\pi_{\theta}}(s,a) \left[R_{\text{proxy}}(s,a) - \frac{\lambda}{\sqrt{\chi^2(\mu_{\pi_{\theta}} \parallel \mu_{\pi_{\text{ref}}})}} \cdot \frac{\mu_{\pi_{\theta}}(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)} \right]. \quad (37)$$

1179 **Interpretation.** The policy gradient consists of two terms:

- 1180 • A standard term encouraging the policy to increase $R_{\text{proxy}}(s,a)$.
- 1181 • A correction term that penalizes deviations from the reference occupancy $\mu_{\pi_{\text{ref}}}$, while also
- 1182 adjusting for alignment with the proxy reward.

1183 This correction enforces robustness to potential reward hacking by optimizing against adversarially
1184 misaligned interpretations of the proxy reward.

1185 Notice that our derived policy gradient in Equation 36 shares structural similarities with ORPO but
1186 is rooted in a formal robust optimization framework. Unlike ORPO, our formulation introduces
1187 an additional correction term involving both the occupancy ratio and the expected proxy reward,
1188 capturing how the proxy is aligned with the current policy’s behavior. This structure more explicitly
1189 penalizes the combination of distributional shift and proxy overoptimization, discouraging policies
1190 from exploiting proxy-specific artifacts. Both methods share the goal of improving robustness, but
1191 our approach is derived from first principles by directly optimizing for worst-case performance over a
1192 correlation-constrained uncertainty set.

1193 E Additional Implementation Details

1194 E.1 Training Discriminator Network

1195 A core step in our Max-Min optimization algorithm and ORPO is to estimate the Radon-Nikodym
1196 derivative $L(s,a)$, which is critical for computing the χ^2 divergence, as detailed in Appendix E.2. To
1197 this end, we follow prior works [25, 34, 35] and train a discriminator network. Specifically, we sample
1198 a batch of trajectories $D_{\pi_{\text{ref}}}$ from the reference policy π_{ref} and another batch D_{π} from the current
1199 policy π . The batch sizes used for each are specified in Table 2. And then we use a discriminator
1200 architecture identical to that in [25], denoted by $d_{\phi}(s,a)$, which is optimized according to:

$$\begin{aligned} \phi &= \arg \min_{\phi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [\log(1 + e^{d_{\phi}(s,a)})] + \mathbb{E}_{\mu_{\pi}} [\log(1 + e^{-d_{\phi}(s,a)})] \\ &\approx \arg \min_{\phi} \mathbb{E}_{D_{\pi_{\text{ref}}}} [\log(1 + e^{d_{\phi}(s,a)})] + \mathbb{E}_{D_{\pi}} [\log(1 + e^{-d_{\phi}(s,a)})] \end{aligned} \quad (38)$$

1201 It is known that the optimal discriminator satisfies $d^*(s,a) = \log \frac{\mu_{\pi}(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)}$ and we estimate $L(s,a)$
1202 as $\tilde{L}(s,a) = e^{d_{\phi}(s,a)}$ with $d_{\phi}(s,a) \approx d^*(s,a)$. However, in the original ORPO implementation², the
1203 discriminator is not fully optimized during policy learning. Specifically, the discriminator receives
1204 only a small number of gradient updates per reinforcement learning iteration, resulting in underfitting
1205 and inaccurate estimates of the Radon-Nikodym derivative $L(s,a)$.

1206 This undertraining is evident in Figure 2, which shows the discriminator loss across RL iterations.
1207 The loss remains nearly constant (e.g., around 1.4 in the Traffic environment, which is the initial loss
1208 value as shown in Figure 3a), indicating that the discriminator is not learning effectively. This limits
1209 its ability to distinguish between π and π_{ref} , especially for state-action pairs where their occupancy
1210 distributions diverge.

1211 To address this, we substantially increase the number of gradient updates per iteration and carefully
1212 tune the learning rate. Our goal is to strike a practical balance between training time and discriminator
1213 quality: while fully training the discriminator to convergence each iteration is computationally
1214 expensive, insufficient training leads to inaccurate divergence estimates and unstable optimization.

1215 Figure 3 shows that in our implementation, the discriminator loss consistently decreases within each
1216 iteration, e.g., from an initial value around 1.4 to below 0.2 in the Traffic environment, indicating

²<https://github.com/cassidylaidlaw/orpo/tree/main>

1217 effective optimization and more accurate occupancy-ratio estimation. In the Glucose and Pandemic
 1218 environments, however, we observe that training the discriminator for too long leads to slower
 1219 convergence and little improvement in loss. In these cases, we apply early stopping to limit training
 1220 time. The specific training schedules are provided in Table 2.

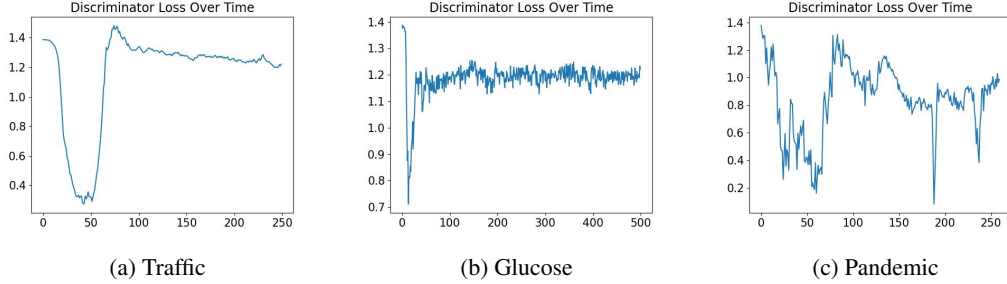


Figure 2: Discriminator loss across RL iterations in the **original ORPO implementation**. The loss stays flat and high (~ 1.4 for the traffic environment), indicating the discriminator is not adequately trained. This undermines the accuracy of the estimated occupancy ratios.

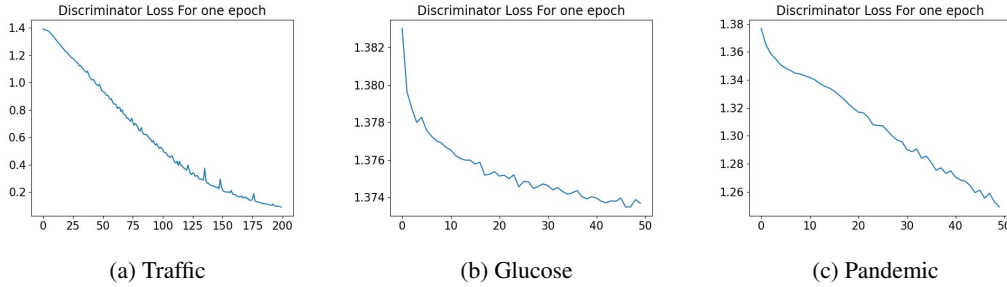


Figure 3: Discriminator loss over training steps *within each RL iteration* in our implementation. The loss decreases rapidly from its initial value (e.g., 1.4 to values near 0.2 in the Traffic environment), indicating successful training and improved accuracy of occupancy-ratio estimates.

1221 As for the discriminator network architecture, we follow the same structure described in [25]. For each
 1222 environment, we employ a fully connected neural network with two hidden layers, each consisting
 1223 of 256 units and ReLU activations. Table 2 summarizes the hyperparameters used for discriminator
 1224 training across different environments.

Table 2: Hyperparameters used for discriminator network training across different environments.

Hyperparameter	Traffic	Glucose	Pandemic
Learning rate	5×10^{-3}	1×10^{-2}	5×10^{-4}
SGD epochs per iteration	200	20	15
Batch size	40000	100000	3860
SGD minibatch size	16384	1024	64

1225 E.2 Derivation of Max-Min Policy Optimization

1226 Using the estimated $d_\phi(s, a)$ from trained discriminator as discussed in Appendix E.1, we can compute
 1227 the χ^2 divergence via:

$$\chi^2(\mu_\pi \parallel \mu_{\pi_{\text{ref}}}) = \mathbb{E}_{\mu_\pi} \left[\frac{\mu_\pi(s, a)}{\mu_{\pi_{\text{ref}}}(s, a)} - 1 \right] \approx \mathbb{E}_{D_\pi} \left[e^{d_\phi(s, a)} - 1 \right]. \quad (39)$$

1228 For environments where both state and action spaces are discrete, we directly estimate the occupancy
 1229 measure via empirical sampling. Specifically, given the same batch of trajectories D collected from

1230 policy π (as used for training the discriminator), we approximate the discounted occupancy measure
 1231 as follows [55, 56]:

$$\tilde{\mu}_\pi^D(s, a) := (1 - \gamma) \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^T \gamma^t \mathbb{I}\{s_t^i = s, a_t^i = a\}, \quad (40)$$

1232 where $\mathbb{I}\{\cdot\}$ is the indicator function. Using this empirical estimate, we can compute the Radon-
 1233 Nikodym derivative and χ^2 divergence without training a discriminator.

1234 In our formulation, we assume that the proxy reward is normalized with respect to the reference policy
 1235 π_{ref} . To achieve this, we reuse the same batch of trajectories $D_{\pi_{\text{ref}}}$ sampled from π_{ref} to estimate the
 1236 expected return $\tilde{J}(\pi_{\text{ref}}, R_{\text{proxy}})$ using:

$$\tilde{J}(\pi_{\text{ref}}, R_{\text{proxy}}) = (1 - \gamma) \frac{1}{N} \sum_{i=1}^N R_{\text{proxy}}(\tau^{(i)}) \quad (41)$$

1237 where each $\tau^{(i)} = (s_0, a_0, s_1, a_1, \dots, s_T) \sim D_{\pi_{\text{ref}}}$ is a trajectory sampled from π_{ref} , N is the number
 1238 of sampled trajectories, and $R_{\text{proxy}}(\tau^{(i)}) = \sum_{t=0}^T \gamma^t R_{\text{proxy}}(s_t^{(i)}, a_t^{(i)})$. This estimation is unbiased
 1239 when using trajectories generated by the policy π . To estimate the empirical variance of the proxy
 1240 reward, we use:

$$\tilde{\sigma}_{R_{\text{proxy}}}^2 = \tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^2[R_{\text{proxy}}] - \tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}]^2 \quad (42)$$

1241 We estimate $\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}^2]$ using:

$$\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}[R_{\text{proxy}}^2] = (1 - \gamma) \frac{1}{N} \sum_{i=1}^N R_{\text{proxy}}^2(\tau^{(i)})$$

1242 where $R_{\text{proxy}}^2(\tau^{(i)}) = \sum_{t=0}^T \gamma^t R_{\text{proxy}}^2(s_t^{(i)}, a_t^{(i)})$. However, estimating $\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^2[R_{\text{proxy}}]$ directly from a
 1243 single batch introduces bias, because the square of an empirical mean is not an unbiased estimator of
 1244 the square of the true mean. To obtain an unbiased estimate of $\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^2[R_{\text{proxy}}]$, we apply the double-
 1245 sampling technique [57, 58]. Specifically, we independently sample another batch of trajectories,
 1246 denoted $D_{\pi_{\text{ref}}}^*$, from the reference policy π_{ref} , and compute:

$$\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^2[R_{\text{proxy}}] = \tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^{D_{\pi_{\text{ref}}}}[R_{\text{proxy}}] \times \tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^{D_{\pi_{\text{ref}}}^*}[R_{\text{proxy}}]$$

1247 where $\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^{D_{\pi_{\text{ref}}}}[R_{\text{proxy}}]$ and $\tilde{\mathbb{E}}_{\mu_{\pi_{\text{ref}}}}^{D_{\pi_{\text{ref}}}^*}[R_{\text{proxy}}]$ denote the empirical returns computed from the two indepen-
 1248 dent batches using Equation 41. This ensures an unbiased estimation of $\mathbb{E}_{\mu_{\pi}}^2[R_{\text{proxy}}]$, which is critical
 1249 for correctly computing the regularization term in the objective 27.

1250 We then normalize the proxy reward for each state-action pair in D_π as:

$$R_{\text{proxy}}^{\text{norm}}(s, a) = \frac{R_{\text{proxy}}(s, a) - \tilde{J}(\pi_{\text{ref}}, R_{\text{proxy}})}{\tilde{\sigma}_{R_{\text{proxy}}}} \quad (43)$$

1251 For notational simplicity, we will continue to use R_{proxy} to denote the normalized proxy reward
 1252 throughout the remainder of this section.

1253 We use the same batch of sampled trajectories D_π from current policy π to estimate $\mathbb{E}_{\mu_\pi}[R_{\text{proxy}}]$
 1254 using:

$$\tilde{\mathbb{E}}_{\mu_\pi}[R_{\text{proxy}}] = \tilde{J}(\pi, R_{\text{proxy}}) = (1 - \gamma) \frac{1}{N} \sum_{i=1}^N R_{\text{proxy}}(\bar{\tau}^{(i)}) \quad (44)$$

1255 where each $\bar{\tau}^{(i)} = (s_0, a_0, s_1, a_1, \dots, s_T) \sim D_\pi$ is a trajectory sampled from π . To estimate
 1256 $\tilde{\mathbb{E}}_{\mu_\pi}^2[R_{\text{proxy}}]$, we apply the same double-sampling technique. Specifically, we independently sample
 1257 another batches of trajectories, denoted D_π^* , from the current policy π , and compute:

$$\tilde{\mathbb{E}}_{\mu_\pi}^2[R_{\text{proxy}}] = \tilde{\mathbb{E}}_{\mu_\pi}^{D_\pi}[R_{\text{proxy}}] \times \tilde{\mathbb{E}}_{\mu_\pi}^{D_\pi^*}[R_{\text{proxy}}], \quad (45)$$

1258 where $\tilde{\mathbb{E}}_{\mu_\pi}^{D_\pi}[R_{\text{proxy}}]$ and $\tilde{\mathbb{E}}_{\mu_\pi}^{D_\pi^*}[R_{\text{proxy}}]$ denote the empirical returns computed from the two independent
 1259 batches using Equation 44.

1260 Putting all the steps together, the maxmin algorithm is in Algorithm 2:

Algorithm 2 Max-Min Policy Optimization

```

1: Initialize policy parameters  $\theta$ 
2: Initialize reference policy  $\pi_{\text{ref}}$  and collect trajectories  $D_{\pi_{\text{ref}}}$ 
3: Estimate  $J(\pi_{\text{ref}}, R_{\text{proxy}})$  using Equation 41 and  $\sigma_{R_{\text{proxy}}}^2$  using Equation 42
4: for each iteration do
5:   Sample trajectories  $D_\pi$  from current policy  $\pi_\theta$ 
6:   if discrete environment then
7:     Estimate occupancy measure using Equation 40
8:   else
9:     Train discriminator  $d_\phi$  by minimizing Equation 38
10:  end if
11:  Estimate  $\chi^2$  divergence using Equation 39
12:  Normalize proxy reward for each state-action pair in  $D_\pi$  using Equation 43
13:  Estimate proxy reward expectation  $\mathbb{E}_{\mu_\pi}[R_{\text{proxy}}]$  using Equation 44
14:  Estimate  $\mathbb{E}_{\mu_\pi}^2[R_{\text{proxy}}]$  via double-sampling using Equation 45
15:  Update policy  $\pi_\theta$  using PPO to maximize robust objective in Equation 27
16: end for

```

E.3 Derivation of Linear Max-Min Policy Optimization

As for the Linear Max-Min optimization problem, following the discussion in D.4–D.7, we first need to estimate Q :

$$Q = \sum_{(s,a)} C(s,a) \phi(s,a) \phi(s,a)^\top$$

where $C(s,a) = \mu_{\pi_{\text{ref}}}(s,a)$, $\phi(s,a)$ is a vector of known feature functions. Define

$$\bar{d}_\phi(s,a) = \log \frac{\mu_{\pi_{\text{ref}}}(s,a)}{\mu_\pi(s,a)}$$

we can rewrite Q via importance sampling [59]:

$$Q = \mathbb{E}_{\mu_\pi} \left[e^{\bar{d}_\phi(s,a)} \phi(s,a) \phi(s,a)^\top \right]$$

Note that $\bar{d}_\phi(s,a)$ differs slightly from $d_\phi(s,a) = \log \frac{\mu_\pi(s,a)}{\mu_{\pi_{\text{ref}}}(s,a)}$ used in the Max-Min optimization algorithm discussed in Appendix E.2, where the numerator and denominator are reversed. To estimate $\bar{d}_\phi(s,a)$, we again train a similar discriminator network as described in Appendix E.1 by minimizing:

$$\begin{aligned} \phi &= \arg \min_{\phi} \mathbb{E}_{\mu_{\pi_{\text{ref}}}} [\log(1 + e^{-d_\phi(s,a)})] + \mathbb{E}_{\mu_\pi} [\log(1 + e^{d_\phi(s,a)})] \\ &\approx \arg \min_{\phi} \mathbb{E}_{D_{\pi_{\text{ref}}}} [\log(1 + e^{-d_\phi(s,a)})] + \mathbb{E}_{D_\pi} [\log(1 + e^{d_\phi(s,a)})] \end{aligned} \quad (46)$$

At optimality, the discriminator satisfies:

$$\bar{d}^*(s,a) = \log \frac{\mu_{\pi_{\text{ref}}}(s,a)}{\mu_\pi(s,a)}$$

And we use $\bar{d}_\phi(s,a) \approx \bar{d}^*(s,a)$. We then estimate Q :

$$\tilde{Q} = (1 - \gamma) \mathbb{E}_{D_\pi} \left[\sum_{t=0}^{\infty} \gamma^t e^{\bar{d}_\phi(s_t, a_t)} \phi(s_t, a_t) \phi(s_t, a_t)^\top \right] \quad (47)$$

We then perform feature whitening by applying a linear transformation:

$$\tilde{\phi}(s,a) = \tilde{W} \phi(s,a)$$

where $\tilde{W} = \tilde{Q}^{-1/2}$ is the matrix square root inverse of \tilde{Q} .

All subsequent quantities are computed using the transformed features $\tilde{\phi}$. As before, we also normalize the proxy reward for each state-action pair in D_π using Equation 43.

After whitening, we need to estimate the gradient of each dual variables $(\lambda_1, \lambda_2, \lambda_3)$ as derived in Appendix D.7.

1277 **Estimating $C_{\phi,j}$ and $D_{\phi,j}$.** Recall that

$$C_{\phi,j} = \sum_{(s,a)} C(s,a) \phi_j(s,a)$$

1278 which can be rewritten via importance sampling as:

$$C_{\phi,j} = \mathbb{E}_{\mu_\pi} \left[e^{\bar{d}_\phi(s,a)} \tilde{\phi}_j(s,a) \right]$$

1279 and then can be approximated via:

$$\tilde{C}_{\phi,j} = (1 - \gamma) \mathbb{E}_{D_\pi} \left[\sum_{t=0}^{\infty} \gamma^t e^{\bar{d}_\phi(s_t, a_t)} \tilde{\phi}_j(s_t, a_t) \right]$$

1280 Similarly, recall that

$$D_{\phi,j} = \sum_{(s,a)} D(s,a) \phi_j(s,a)$$

1281 where $D(s,a) = \mu_{\pi_{\text{ref}}}(s,a) \cdot R_{\text{proxy}}(s,a)$. Using importance sampling, we can write:

$$D_{\phi,j} = \mathbb{E}_{\mu_\pi} \left[e^{\bar{d}_\phi(s,a)} R_{\text{proxy}}(s,a) \tilde{\phi}_j(s,a) \right]$$

1282 and can be approximated in practice by:

$$\tilde{D}_{\phi,j} = (1 - \gamma) \mathbb{E}_{D_\pi} \left[\sum_{t=0}^{\infty} \gamma^t e^{\bar{d}_\phi(s_t, a_t)} R_{\text{proxy}}(s_t, a_t) \tilde{\phi}_j(s_t, a_t) \right]$$

1283 **Estimating $q_j(\lambda_1, \lambda_2)$.** Recall that

$$\begin{aligned} q_j(\lambda_1, \lambda_2) &= \sum_{(s,a)} (v(s,a) - \lambda_1 D(s,a) - \lambda_2 C(s,a)) \phi_j(s,a) \\ &= \sum_{(s,a)} v(s,a) \phi_j(s,a) - \lambda_1 \sum_{(s,a)} D(s,a) \phi_j(s,a) - \lambda_2 \sum_{(s,a)} C(s,a) \phi_j(s,a) \\ &= \mathbb{E}_{\mu_\pi} [\phi_j(s,a)] - \lambda_1 D_{\phi,j} - \lambda_2 C_{\phi,j} \end{aligned}$$

1284 where $v(s,a) = \mu_\pi(s,a)$ and $\mathbb{E}_{\mu_\pi}[\phi_j(s,a)]$ is the discounted feature expectation under the policy π .
1285 We can estimate the first term using:

$$\tilde{\mathbb{E}}_{\mu_\pi} [\phi_j(s,a)] = (1 - \gamma) \frac{1}{N} \sum_{i=1}^N \tilde{\phi}_j(\bar{\tau}^i)$$

1286 where each $\bar{\tau}^{(i)} = (s_0, a_0, s_1, a_1, \dots, s_T) \sim D_\pi$ is a trajectory sampled from π , and $\tilde{\phi}_j(\bar{\tau}^{(i)}) =$
1287 $\sum_{t=0}^T \gamma^t \tilde{\phi}_j(s_t^{(i)}, a_t^{(i)})$. Given the above estimates, we can finally compute:

$$\tilde{q}_j(\lambda_1, \lambda_2) = \tilde{\mathbb{E}}_{\mu_\pi} [\phi_j(s,a)] - \lambda_1 \tilde{D}_{\phi,j} - \lambda_2 \tilde{C}_{\phi,j}$$

1288 With the above estimation, we can compute the gradient and solve for the optimal dual variables
1289 $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ using the Levenberg-Marquardt algorithm [60], a damped least-squares method
1290 designed for solving nonlinear systems of equations. Specifically, we use the root solver in SciPy [61]
1291 to find the stationary point of the gradient $\nabla_\lambda g(\lambda) = 0$. We initialize the optimization with $\lambda_1 = 0$,
1292 $\lambda_2 = 0$, and $\lambda_3 = -1$, and enforce $\lambda_3 < 0$ throughout training to ensure concavity of the dual
1293 objective $g(\lambda)$. To enforce the non-negativity constraint $\theta \geq 0$ as required by the analytical form in
1294 Equation 33, we manually clip each θ_i to ensure it remains non-negative. Future work may explore
1295 alternative solvers better suited to constrained optimization.

1296 Recall that the optimal primal variable θ_j^* is:

$$\theta_j^*(\lambda) = \max \left(0, \frac{q_j(\lambda_1, \lambda_2)}{2\lambda_3} \right)$$

Algorithm 3 Linear Max-Min Policy Optimization

```
1: Initialize policy parameters  $\theta$ 
2: Initialize reference policy  $\pi_{\text{ref}}$  and collect trajectories  $D_{\pi_{\text{ref}}}$ 
3: Estimate  $J(\pi_{\text{ref}}, R_{\text{proxy}})$  using Equation 41 and  $\sigma_{R_{\text{proxy}}}^2$  using Equation 42
4: for each iteration do
5:   Sample trajectories  $D_\pi$  from current policy  $\pi_\theta$ 
6:   if discrete environment then
7:     Estimate occupancy measure using Equation 40
8:   else
9:     Train discriminator  $d_\phi$  by minimizing Equation 46
10:  end if
11:  Normalize proxy reward for each state-action pair in  $D_\pi$  using Equation 43
12:  Estimate  $\tilde{Q}$  using Equation 47
13:  Compute feature transformation  $\tilde{W} = \tilde{Q}^{-1/2}$  and transform features  $\tilde{\phi}(s, a) = \tilde{W}\phi(s, a)$ 
14:  Estimate  $\tilde{C}_{\phi,j}, \tilde{D}_{\phi,j}$  using transformed features
15:  Compute  $\tilde{q}_j(\lambda_1, \lambda_2)$  for all features
16:  Solve for optimal dual variables  $(\lambda_1, \lambda_2, \lambda_3)$  by maximizing the dual objective (Equation 34)
17:  Compute the optimal primal variable  $\tilde{\theta}_j^*$  for all features
18:  Update policy  $\pi_\theta$  using PPO to maximize the robust objective in Equation 28
19: end for
```

1297 After optimizing for the dual variables, we can substitute the optimal $(\lambda_1^*, \lambda_2^*, \lambda_3^*)$ back into the above
1298 equation and get:

$$\tilde{\theta}_j^*(\lambda) = \max \left(0, \frac{\tilde{q}_j(\lambda_1^*, \lambda_2^*)}{2\lambda_3^*} \right)$$

1299 for all features. Then we can substitute the optimal $\tilde{\theta}^*$ back in the robust reward objective in
1300 Equation 28 and train the policy π to maximize the outer problem using the standard reinforcement
1301 learning algorithm proximal policy optimization (PPO) [33].

1302 Putting all the steps together, the linear maxmin algorithm is in Algorithm 3:

1303 E.4 Environment Description and Reward Hacking Types

1304 **Traffic.** This environment simulates a highway merging scenario, adapted from [36, 62, 21], where
1305 a group of autonomous vehicles (AVs) controlled by an RL agent must merge into human-driven
1306 traffic. Each AV observes its own state (position and velocity) and those of nearby vehicles, and
1307 outputs continuous acceleration actions. The true reward is designed to ensure smooth and efficient
1308 traffic flow, encouraging low commute times and gentle accelerations. The reference policy π_{ref} is a
1309 behavioral cloning (BC) policy trained on demonstrations generated by the Intelligent Driver Model
1310 (IDM) [63].

1311 **Pandemic.** Based on the PandemicSimulator [24], this environment models infection dynamics
1312 using an extended SEIR model. At each timestep, the agent selects a lockdown policy to control
1313 the spread of disease while minimizing societal costs. The true reward balances infection severity,
1314 political disruption, and policy smoothness over time. The reference policy is trained via behavioral
1315 cloning on a mixture of realistic and hand-crafted policy trajectories.

1316 **Glucose Monitoring.** This environment uses the SimGlucose simulator [64, 22], where an RL
1317 agent administers insulin doses to a simulated patient with Type 1 Diabetes. The goal is to maintain
1318 safe blood glucose levels and minimize long-term health risk. The reference policy is trained via
1319 behavioral cloning using data generated by a PID controller with clinically tuned parameters [65].
1320 Proxy rewards in this setting often reflect surrogate objectives such as treatment cost or patient
1321 burden.

1322 **Tomato Watering GridWorld.** This environment presents a simple spatial grid where the agent
1323 waters tomato plants. The true reward corresponds to the number of tomatoes correctly watered.
1324 However, the proxy reward includes an artificially high bonus at a specific grid location (a “sprinkler

state”), which causes the agent to overfit by remaining in that region despite little actual benefit to overall tomato growth. The reference policy follows [25], with 10% random actions added to allow for policy improvement.

Types of Reward Hacking. We adopt the taxonomy proposed in [36] to classify the kinds of proxy reward misalignments that lead to reward hacking. Our selected environments span all three major categories:

- **Misweighting:** The proxy reward includes all relevant objectives but uses incorrect relative weights. Our Linear Max-Min method specifically seeks the most adversarial weighting in this space.
- **Ontological:** The proxy captures the correct high-level goal using different or incomplete features. In the **Traffic** environment, the true reward combines commute time, acceleration, and headway, whereas the proxy replaces commute time with velocity. In the **Pandemic** environment, the true reward penalizes infections, political cost, lower stage changes, and non-smooth policies, while the proxy omits the political cost entirely. Similarly, in **Glucose**, the proxy reward only considers the expected patient costs while the true reward only measures the health risk.
- **Scope:** The proxy evaluates behavior over a limited domain. In the **Tomato** environment, the true reward reflects the number of tomatoes successfully watered. However, the proxy introduces a large bonus at a specific state (the sprinkler), incentivizing the agent to pursue this location at the expense of fulfilling the intended watering task.

E.5 Additional Experiment Setup

For the policy networks, we follow the architectures described in [25]. In the Pandemic, Traffic, and Tomato environments, we use fully connected neural networks with 2 layers of 128 units, 4 layers of 512 units, and 4 layers of 512 units, respectively. For the Glucose environment, we employ a three-layer LSTM network, where each LSTM layer has 64 units. We use the pre-trained policies provided in the ORPO repository³ as the reference policies π_{ref} . We initialize the policy network with the corresponding pre-trained checkpoint for the Traffic, Glucose, and Pandemic environments, and initialize a random policy for the Tomato environment. Table 3 summarizes the hyperparameters used for PPO training across all models and environments.

Table 3: Hyperparameters used for PPO training across different environments.

Hyperparameter	Traffic	Glucose	Pandemic	Tomato
Training iterations	250	500	260	500
Batch size	40000	100000	3860	3000
Optimizer	Adam	Adam	Adam	Adam
Learning rate	5×10^{-5}	1×10^{-5}	0.0003	1×10^{-3}
Gradient clipping	N/A	10	10	0.1
Discount factor (γ)	0.99	0.99	0.99	0.99
Random seed	0	0	0	0
GAE coefficient (λ)	0.97	0.98	0.95	0.98
Entropy coefficient (start)	0.01	0.01	0.1	0.01
Entropy coefficient (end)	0.01	0.01	0.01	0.01
KL target	0.02	1×10^{-3}	0.01	1×10^{-3}
Value function loss clipping	10000	100	20	10
Value function loss coefficient	0.5	0.0001	0.5	0.1
Share value function layers	True	True	True	False

As for the reward used during training and evaluation, we follow the same setup as ORPO [25]. All policies are trained using **only the proxy reward**, while both the true and proxy rewards are used for evaluation.

³https://github.com/cassidylaidlaw/orpo/tree/main/data/base_policy_checkpoints

In the **Traffic** environment, the proxy reward is a weighted combination of *velocity*, *acceleration*, and *headway*, with weights 1, 1, and 0.1, respectively. The true reward, on the other hand, uses *commute time*, *acceleration*, and *headway*, also weighted 1, 1, and 0.1.

In the **Pandemic** environment, the proxy reward is composed of *infection summary absolute*, *lower stage*, and *smooth stage changes*, with weights 10, 0.1, and 0.01. The true reward adds a *political* component to these three features and is weighted with 10.

For the **Glucose** environment, the proxy reward includes only one feature: *expected patient cost*. The true reward is based on *magni_bg*, which measures the health risk of the patient.

In the **Tomato** environment, the true reward counts the number of *watered tomato*. The proxy reward adds a large bonus at a specific state (*sprinkler*), incentivizing the agent to reach that location regardless of its impact on the primary task.

For our Max-Min and Linear Max-Min policy optimization algorithms, the correlation parameter r serves as an additional hyperparameter. In practice, as with ORPO [25], r may only be approximately estimated, and there is currently no principled method for selecting its optimal value. To address this, we perform a grid search over $r \in \{0.1, 0.2, \dots, 0.9\}$ for each environment and measure the resulting Max-Min and Linear Max-Min policy expected returns under the worst-case or linear worst-case reward. Additional analysis of how different training values of r affect robustness under varying evaluation r values is provided in Appendix F.2. Unless otherwise noted, we use the following r values for training and evaluation throughout our experiments: $r = 0.3$ for **Traffic**, $r = 0.7$ for **Pandemic**, $r = 0.9$ for **Glucose**, and $r = 0.4$ for **Tomato**. As for ORPO policy, we trained with occupancy-measure χ^2 regularization, using the official implementation from [25]. All hyperparameters are set as recommended to ensure optimal performance. The ORPO* shares the exact same setting as the ORPO policy with the full discriminator training schedule as in our algorithms.

Evaluation of the worst-case performance. Theoretically, in the absence of structural constraints on the reward function—as opposed to the case of linear rewards—the worst-case reward of a policy in state-action pairs unvisited by π_{ref} can be arbitrarily negative without violating the correlation constraint. However, assigning extremely negative values is impractical in real-world scenarios due to domain constraints. Moreover, doing so would render all policies with at least one unseen state-action pair equally poor in terms of worst-case reward, obscuring meaningful comparisons. To address this, we define a minimal feasible reward value R_{\min} and assign it to all unseen state-action pairs. The actual expected worst-case reward (**Worst***) is thus calculated as:

$$\sum_{(s,a):\mu_{\pi_{\text{ref}}}(s,a)>0} \mu_{\pi}(s,a)R_{\text{worst}} + \sum_{(s',a'):\mu_{\pi_{\text{ref}}}(s',a')=0} \mu_{\pi}(s',a')R_{\min}$$

where the first part is derived from the adversarial reward function given by our inner minimization solution, and the second part applies to state-action pairs unvisited by π_{ref} .

In practice, however, environments like Traffic, Pandemic, and Glucose are continuous with large state-action spaces, making it difficult to reliably estimate $\mu_{\pi}(s,a)$ and $\mu_{\pi_{\text{ref}}}(s,a)$ from a limited number of trajectories. As a result, identifying unvisited or low-density regions in these environments is far more ambiguous. Therefore, for these continuous environments, we rely on the output of the discriminator as a signal for detecting unseen state-action pairs. Specifically, if the discriminator outputs infinity (or diverges numerically) for a given state-action, we treat this as an indication that the state-action was never visited by the reference policy π_{ref} . We approximate the total occupancy (**Occ**) over such state-action pairs by computing their frequency in the sampled trajectories, and use the expected worst-case reward (**Worst**) of a policy π over the remaining state-action pairs as the default worst-case performance metric: $\sum_{(s,a):d_{\phi}(s,a)<\infty} \mu_{\pi}(s,a)R_{\text{worst}}(s,a)$. In contrast, for the discrete Tomato environment, we directly estimate the occupancy measure by sampling state-action pairs and then compute **Worst*** accordingly. Further details on this procedure are provided in Appendix F.2.

To compare the worst-case performance of different policies, we sample 200 trajectories in the **Traffic** and **Glucose** environments, 20 trajectories in **Pandemic**, and 1000 trajectories in **Tomato** to estimate the worst-case performance of a policy.

Evaluation of policy robustness. To evaluate robustness across different correlation levels, we uniformly sample candidate vectors θ , where each component θ_i is drawn from the interval $[0, 1]$. We use the same number of trajectories sampled from the reference policy π_{ref} to determine whether

1408 it satisfies the correlation constraint:

$$\mathbb{E}_{\mu_{\pi_{\text{ref}}}} \left[\frac{\boldsymbol{\theta}^\top \boldsymbol{\phi} - M}{V} \cdot R_{\text{proxy}} \right] = r,$$

1409 where M and V denote the mean and standard deviation of $\boldsymbol{\theta}^\top \boldsymbol{\phi}$ under the reference policy.

1410 **Note:** For our worst-case performance evaluation, we explicitly normalize the reward to have zero
 1411 mean and unit variance under the reference policy (enforcing $M = 0$ and $V = 1$). In contrast, for the
 1412 robustness evaluation across correlation levels, we do not apply such normalization. This allows us to
 1413 report the average reward under each $\boldsymbol{\theta}$ in its original scale, reflecting variability more comparable to
 1414 the original true reward landscape.

1415 E.6 Training Time and Complexity

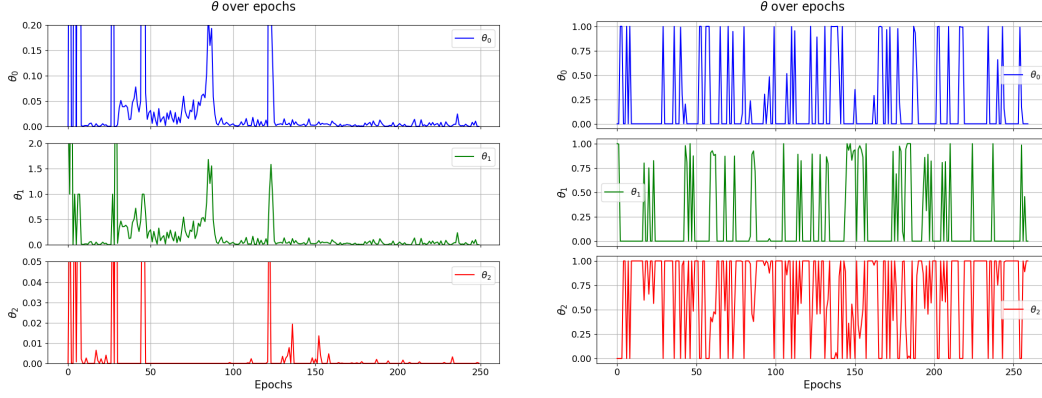
1416 Table 4 reports the total training time for each algorithm across different environments. All experi-
 1417 ments were conducted on a single NVIDIA RTX 4090 GPU (24GB memory) and a 13th Gen Intel
 1418 Core i9-13900KF CPU (32 threads). We implemented all methods in Python 3.9 using PyTorch
 1419 2.6.0 [66] and RLlib [67].

1420 The training times across different environments are summarized in Table 4. Since the training
 1421 durations for ORPO*, Max-Min, and Linear Max-Min differ by less than one hour in each setting,
 1422 we group them together for brevity. As shown in the table, all three methods require more training
 1423 time compared to the original ORPO implementation. The increased training time primarily results
 1424 from additional gradient steps used to more thoroughly train the discriminator network. Specifically,
 1425 the per-iteration training time is approximately 2.5 minutes for Traffic, 4.6 minutes for Pandemic,
 1426 and 8.9 minutes for Glucose. This leads to a total training time increase from roughly 7 hours to
 1427 37 hours in Glucose. However, the added cost is environment-dependent and remains moderate in
 1428 simpler settings, for example, increasing from 5 to 10 hours in Traffic. These results demonstrate our
 1429 design objective of achieving a practical trade-off between computational overhead and the quality of
 1430 divergence estimation.

Table 4: Approximate training time for each algorithm across different environments.

Algorithm	Traffic	Glucose	Pandemic	Tomato
ORPO	~5h	~7h	~14h	~1h
ORPO* / Max-Min / Linear Max-Min	~10h	~37h	~19h	~1h

1431 **Complexity.** At first glance, regularization-based approaches like ORPO may appear more compu-
 1432 tationally efficient than max-min optimization, which often involves iterative procedures to solve
 1433 both inner and outer objectives. However, in practice, ORPO requires repeatedly estimating the χ^2
 1434 divergence between policy distributions during each policy update step. This estimation is done
 1435 by training a discriminator network, which itself involves multiple optimization steps per iteration.
 1436 In contrast, our Max-Min formulation admits a closed-form solution for the inner minimization
 1437 over reward functions. This allows us to avoid iterative solving in the inner loop entirely. For the
 1438 Linear Max-Min variant, although a closed-form expression for the dual variables is not avail-
 1439 able, the corresponding dual optimization problem is smooth and well-posed, and can be solved
 1440 efficiently using standard gradient-based methods. Therefore, despite the max-min structure, our
 1441 method does not incur higher practical complexity compared to ORPO. In fact, both approaches rely
 1442 on discriminator-based divergence estimation and perform comparable amounts of computation per
 1443 iteration. The main difference lies in the structure of the objective, not in the asymptotic or empirical
 1444 complexity. In summary, ORPO does not inherently enjoy a complexity advantage over our Max-Min
 1445 or Linear Max-Min algorithms.



(a) Traffic environment: θ_0 (velocity), θ_1 (acceleration), θ_2 (headway). (b) Pandemic environment: θ_0 (infection summary), θ_1 (early-stage), θ_2 (smoothness).

Figure 4: Evolution of adversarial reward weights θ over training epochs for different environments using the Linear Maxmin method.

F Additional Experiment Results

F.1 Feature Weights in Linear Max-Min Optimization during Training

Figure 4 visualizes the evolution of each component of the linear worst-case reward weight vector θ during training in the Traffic and Pandemic environments. We observe distinct behaviors in the dynamics of θ across tasks.

In the Traffic environment (Figure 4a), we observe that the three θ parameters vary significantly in scale. Specifically, θ_1 (acceleration) exhibits the largest magnitude, ranging from 0 to 2, while θ_2 (headway) has the smallest scale, ranging from 0 to 0.05. This highlights how the linear max-min algorithm assigns different levels of penalization to each feature. Moreover, we also observe distinct phases in the dynamics of θ over the course of training. In the early epochs (<50), all components, especially θ_1 (acceleration) and θ_2 (headway), exhibit high-frequency fluctuations. At this point, the dual optimization problem is not yet well-conditioned, and the adversarial reward is highly sensitive to small changes in occupancy or feature values. As training progresses (\sim epochs 100–250), the parameters begin to stabilize. Most notably, θ_2 (headway) converges close to zero and remains suppressed, indicating that the worst-case reward does not emphasize this feature. This may suggest that headway is less harmful under adversarial reweighting compared to others (velocity or accel) or is already well aligned with the reference policy π_{ref} . Meanwhile, θ_1 (acceleration) consistently exhibits higher values and sharper spikes than the other components. This indicates that acceleration plays a dominant role in the adversarial reward, likely because policies that optimize for the proxy reward tend to exploit aggressive acceleration patterns that diverge significantly from the behavior of π_{ref} . In contrast, θ_0 (velocity) remains small and relatively stable throughout training, suggesting that speed alone is not strongly penalized under adversarial interpretations.

Overall, the observed pattern reflects the interpretability and sparsity benefits of the linear max-min formulation. The model is able to selectively emphasize features that are most vulnerable to reward hacking, while suppressing those that are either irrelevant or well-aligned. This structured behavior supports the practical value of using linearly parameterized worst-case rewards to improve policy robustness.

In the Pandemic environment (Figure 4b), unlike the Traffic environment, where θ converged to a sparse and interpretable solution, we observe high variability across all components throughout training. In particular, we find the following pattern:

1. **Persistent fluctuations.** All three components exhibit frequent oscillations over the course of 260 epochs. This ongoing instability suggests that the adversarial reward continually adapts as the policy changes, likely due to the environment’s temporal sensitivity and complex dynamics.

2. θ_2 (**smoothness**) **remains active**. The smoothness-related component θ_2 is frequently non-zero and relatively stable compared to the others. This indicates that the worst-case reward consistently emphasizes penalizing erratic or unstable responses in the infection trajectory — a behavior often neglected by naive proxy metrics.
3. θ_1 (**early-stage transitions**) **is highly volatile**. The component associated with early infection stage changes spikes intermittently. This suggests that early-stage mismanagement is a recurring vulnerability in the learned policy that the adversarial reward seeks to exploit.
4. θ_0 (**overall infection**) **activates intermittently**. Although θ_0 sometimes spikes, it does not dominate the adversarial reward. This may indicate that the learned policy already accounts for infection magnitude reasonably well, or that smoothness and early-stage control offer more leverage for reward hacking under the proxy constraint.

Overall, this pattern highlights that in more dynamic and temporally complex environments like Pandemic, the worst-case reward remains non-sparse and adapts to different policy weaknesses throughout training. In contrast to the Traffic environment, adversarial emphasis here is broader and more reactive.

F.2 Additional Worst-Case Performance Results

Table 5: Evaluation results on Glucose and Tomato environments. All policies are trained using **only the proxy reward**. In Glucose, the proxy uses *expected patient cost*, and the true reward uses *magni_bg*. In Tomato, the proxy includes *number of watered tomatoes* plus a bonus at a specific state (sprinkler), while the true reward only measures *watered tomatoes*. **Occ** in the Tomato environment denotes total occupancy over state-action pairs unseen by π_{ref} , based on 1000 sampled trajectories. **Worst** refers to the expected worst-case reward computed while excluding those unseen state-action pairs. **Worst*** denotes the actual expected worst-case reward, while R_{\min} represents the minimum possible reward of any state-action pair. All rewards are normalized according to the reference policy π_{ref} .

Method	Tomato					Glucose		
	True	Proxy	Worst	Occ ↓	Worst*	True	Proxy	Worst
ORPO	6.22	6.75	-1.48	2.3e-04	$-1.48 + R_{\min} \cdot 2.3e-04$	6017.14	100.68	-27.40
ORPO*	3.90	3.85	-0.97	3.12e-05	$-0.97 + R_{\min} \cdot 3.12e-05$	6300.91	116.41	-8.72
Max-Min	4.48	4.57	-1.41	1.13e-05	$-1.41 + R_{\min} \cdot 1.13e-05$	6287.20	102.57	-1.80

Worst-Case Performance in Glucose and Tomato Environments. Table 5 reports worst-case performance results for the Glucose and Tomato environments. We omit the Linear Max-Min policy from these experiments for the following reasons. In the Glucose environment, both the proxy and true rewards used in prior work [25, 39] are based on a single feature, making the linear reward formulation trivial. Although the original Glucose simulator provides multiple candidate features related to patient health, selecting an appropriate feature combination without prior knowledge of clinical intent is nontrivial. In the Tomato environment, the reward structure is similarly difficult to express in a clean feature-based form suitable for linear modeling. Therefore, in both settings, we report only the results for the Max-Min policy alongside the baselines.

The results for the Glucose and Tomato environments exhibit trends similar to those observed in Traffic and Pandemic (Section 4.2). In the Glucose environment, our Max-Min policy achieves the highest expected worst-case reward, demonstrating better robustness. In contrast, ORPO* appears to outperform others in the Tomato environment in terms of worst-case performance. Recall that these results are reported under **Worst**, the expected worst-case reward restricted to state-action pairs observed under π_{ref} . Since the Tomato environment is discrete, we can explicitly identify which state-action pairs are unseen through sampling, enabling clearer interpretation of their physical meaning as well as the evaluation of the actual worst-case performance **Worst***. The latter corresponds to the **Worst** value plus the product of the occupancy in unseen regions (**Occ**) and R_{\min} . Because Max-Min exhibits the lowest occupancy among all methods, it demonstrates greater robustness under varying assumptions about R_{\min} .

Nevertheless, ORPO* still shows marked improvement over ORPO, both in worst-case return and in reducing occupancy over unseen state-action pairs. As previously noted, in the Glucose environment,

the discriminator fails to detect any state-action pairs missed by the reference policy. This reinforces our earlier concern that the current discriminator training procedures may have limited capacity to identify rare or out-of-distribution events.

Impact of Correlation Parameter Selection on Robustness. In this section, we present additional experiment results to examine how the proxy-true reward correlation parameter r used during training affects the robustness under varying evaluation r .

Table 6: Evaluation of robustness in the Tomato environment across different training-time correlation levels r . **Occ** denotes total occupancy over state-action pairs unseen by π_{ref} , based on 1000 sampled trajectories. **Worst** refers to the expected worst-case reward computed while excluding those unseen state-action pairs.

r	Occ	Worst ($r = 0.1$)	Worst ($r = 0.4$)	Worst ($r = 0.7$)	Worst ($r = 0.9$)
0.1	1.40e-03	-1.36	-1.14	-0.73	-0.26
0.4	1.13e-05	-1.65	-1.41	-0.68	-0.04
0.7	1.07e-02	-2.08	-1.83	-1.31	-0.65
0.9	1.33e-05	-9.06	-8.90	-7.63	-5.51

Table 7: Evaluation of robustness in the Traffic environment across different training-time correlation levels r . **Occ** denotes total occupancy over state-action pairs unseen by π_{ref} , based on 200 sampled trajectories. **Worst** refers to the expected worst-case reward computed while excluding those unseen state-action pairs.

r	Occ	Worst ($r = 0.1$)	Worst ($r = 0.3$)	Worst ($r = 0.7$)	Worst ($r = 0.9$)
0.3	0	-2761.91	-270.84	-81.59	-21.75
0.5	0	-7.64e+04	-1.92e+04	-6095.33	-1327.15
0.9	9.52e-05	-2.96e+05	-9.36e+04	-2.69e+04	-9.19e+03

Table 6 and Table 7 report the robustness evaluation results in the **Tomato** and **Traffic** environments under different training-time values of the correlation parameter r . Several consistent patterns emerge across both environments.

First, for any fixed policy (i.e., fixed training r), we observe that the expected worst-case reward monotonically increases as the evaluation r increases. This aligns with intuition: higher correlation levels correspond to smaller uncertainty sets over rewards, meaning the worst-case reward functions are less adversarial. In contrast, low r values expand the reward uncertainty set, allowing more pathological or implausible reward functions, and thus lead to more pessimistic evaluations.

Second, we find that training with a moderate correlation level, particularly around $r = 0.3$ to 0.4 , yields better robustness across a wide range of evaluation r values. In contrast, training with overly small (e.g., $r = 0.1$ for Tomato) or large (e.g., $r = 0.9$ for Tomato and Traffic) correlation levels degrades robustness. A small r leads to overly conservative training, anticipating extreme forms of reward hacking and thus hurting general performance. On the other hand, a high r overly trusts the proxy reward and fails to hedge against potential deviations, resulting in poor worst-case behavior under reward misspecification. This trade-off highlights that intermediate values of r may strike a better balance between conservativeness and optimism, enabling the policy to generalize to a broader and more plausible spectrum of reward functions.

F.3 Additional Results for Robustness Across Correlation Levels

As discussed previously (Appendix F.2), we do not include linear worst-case evaluation for the Glucose and Tomato environments. Consequently, we cannot perform a uniform search over θ as we do for the Traffic and Pandemic environments (Appendix E.5). As noted in Appendix A, it is generally difficult, and often infeasible, to sample a full reward function over all state-action pairs, particularly in high-dimensional or continuous environments, such as the Glucose environment. To approximate this process for the Tomato environment, which is a discrete environment, we instead

sample 1000 trajectories using the reference policy π_{ref} . We then restrict the search to the visited state-action pairs. For each such pair, we perturb the original proxy reward by adding Gaussian noise with zero mean and variance sampled uniformly from the interval $[0.001, 1]$. We then check whether the resulting perturbed reward \hat{R} satisfies the constraint $\hat{R} \in \mathcal{R}_{\text{corr}}$. As in previous evaluations, we do not explicitly constrain M and V . We sample 20 perturbed reward functions and then use them to evaluate each policy. **Note:** Some policies, such as ORPO, may visit state-action pairs that are not included in the sampled set from π_{ref} . For these unseen state-action pairs, the proxy-true correlation constraint does not apply, as no corresponding reference data is available. In such cases, we default to using the original proxy reward to evaluate those portions of the trajectory.

We emphasize that this procedure is neither optimal nor efficient, and is employed solely for evaluation purposes in the Tomato environment. Designing more principled and scalable methods for reward sampling under correlation constraints remains an important direction for future work.

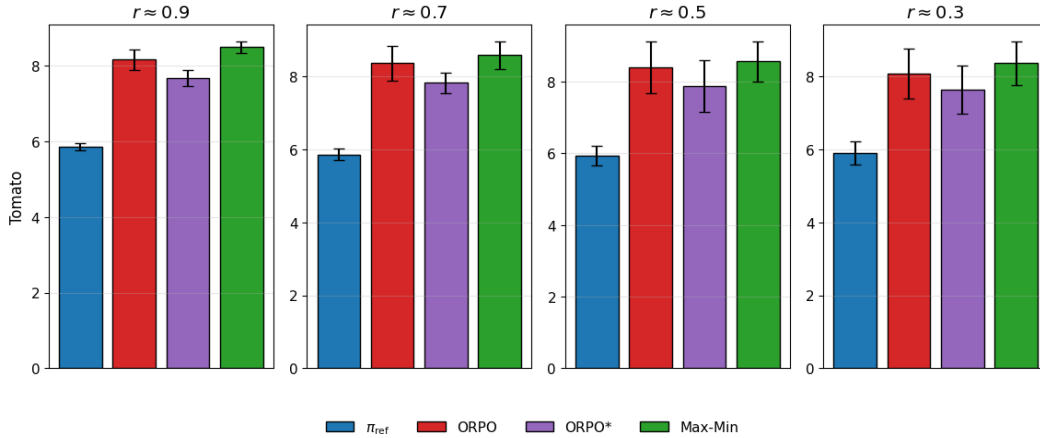


Figure 5: Mean reward and standard deviation under sampled reward functions at different proxy-true reward correlation levels r for the Tomato environment. Our methods (Max-Min) yield higher average performance across all choices of r .

Figure 5 presents the average reward and standard deviation across varying correlation levels r for the Tomato environment. As expected, the reference policy π_{ref} (blue) consistently underperforms across all values of r , though it exhibits the lowest variance—indicating stable yet suboptimal behavior. Interestingly, ORPO* (purple) performs worse than ORPO (red) throughout, suggesting that improving the accuracy of occupancy measure estimation does not necessarily enhance robustness in this environment. In contrast, our Max-Min method (green) achieves the highest average reward across all correlation levels, highlighting its better robustness under reward uncertainty.

F.4 Additional Unnormalized Results

To ensure a fair comparison with prior work [25], which reports results in the unnormalized reward scale, we also include the raw (unnormalized) expected proxy and true rewards. However, for worst-case reward metrics, it is nontrivial to reverse the normalization transformation, as our formulation explicitly constrains the reward to have zero mean and unit variance under the reference policy. Therefore, we omit worst-case results in the unnormalized setting.

Table 8: Unnormalized performance comparison across all environments.

Method	Traffic		Pandemic		Glucose		Tomato	
	True	Proxy	True	Proxy	True	Proxy	True	Proxy
π_{ref}	-1004.33	1474.27	-12.01	-12.01	-79754.84	-117.75	5.96	6.37
ORPO	-665.72	1540.39	-12.74	-10.56	-49669.11	-67.41	9.07	9.07
ORPO*	-798.13	1500.69	-11.01	-11.01	-48250.30	-59.54	7.91	7.91
Max-Min	-751.46	1547.06	-11.08	-11.08	-48318.80	-66.46	8.20	8.20
Linear Max-Min	-677.21	1524.97	-9.92	-5.96	N/A	N/A	N/A	N/A

1573 Table 8 presents the unnormalized performance results across all environments. We observe that
1574 both our Max-Min and Linear Max-Min policies achieve comparable performance to ORPO on most
1575 tasks. Interestingly, the ORPO* variant (with a fully trained discriminator) outperforms the original
1576 ORPO in some environments (e.g., Pandemic and Glucose), but performs worse in others, such as
1577 Traffic and Tomato. While our earlier analysis (Section 4.2) shows that better discriminator training
1578 generally improves worst-case robustness, these results suggest that accurate discriminator estimation
1579 does not always translate to improved performance for every specific reward function. Understanding
1580 the nuanced effects of discriminator optimization on various reward metrics is beyond the scope of
1581 this paper and remains an important direction for future research.